PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(71) Applicant (*for all designated States except US*): EQUATOR TECHNOLOGIES, INC. [US/US]; Suite 900, 520 Pike Street, Seattle, WA 98101–4001 (US).

(72) Inventors; and
(75) Inventors/Applicants (*for US only*): NATARAJAN, Ramachandran [IN/US]; 1771 Warburton Avenue #7, Santa Clara, CA 95050 (US). CAMPBELL, T., George [US/US]; 108 Tynan Way, Portola Valley, CA 94028 (US).

(74) Agents: SANTARELLI, Bryan, A. et al.; Graybeal Jackson Haley LLP, Suite 2460, 777 108th Avenue N.E., Bellevue, WA 98004 (US).
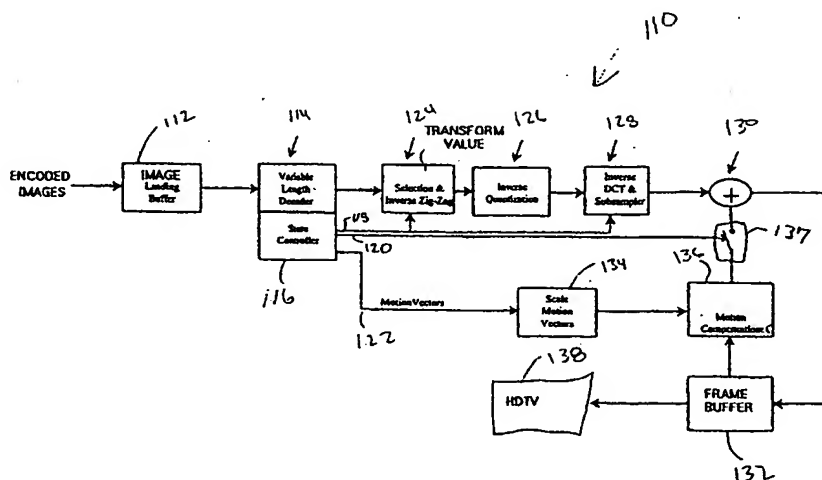
(54) Title: DECODING AN ENCODED IMAGE HAVING A FIRST RESOLUTION DIRECTLY INTO A DECODED IMAGE HAVING A SECOND RESOLUTION

(57) Abstract

An image processing circuit (110) includes a processor that receives an encoded portion (112) of a first version of an image. The processor decodes (114) this encoded portion directly into a decoded portion of a second version of the image, the second version (138) having a resolution that is different than the resolution of the first version. Therefore, such an image processing circuit can decode an encoded hi–res version of an image directly into a decoded lo–res version of the image. Alternatively, the image processing circuit includes a processor that modifies a motion vector (134) associated with a portion of a first version of an image. The processor then identifies a portion of a second image to which the modified motion vector points, the second image having a different resolution than the first version of the first image. Next, the processor generates a portion of a second version of the first image from the identified portion of the second image, the second version of the first image having the same resolution as the second image.

⬤   ⬤

DECODING AN ENCODED IMAGE HAVING A FIRST RESOLUTION DIRECTLY INTO A DECODED IMAGE HAVING A SECOND RESOLUTION

5   *Technical Field:*

  The invention relates generally to image processing circuits and techniques, and more particularly to a circuit and method for decoding an encoded version of an image having a resolution directly into a decoded version of the image having another resolution. For example, such a circuit can down-convert an encoded high-

10 resolution (hereinafter "hi-res") version of an image directly into a decoded low-resolution (hereinafter "lo-res") version of the image without an intermediate step of generating a decoded hi-res version of the image.

Background of the Invention:

15 It is sometimes desirable to change the resolution of an electronic image. For example, an electronic display device such as a television set or a computer monitor has a maximum display resolution. Therefore, if an image has a higher resolution than the device's maximum display resolution, then one may wish to down-convert the image to a resolution that is lower than or equal to the maximum display

20 resolution. For clarity, this is described hereinafter as down-converting a hi-res version of an image to a lo-res version of the same image.

  Figure 1 is a pixel diagram of a hi-res version 10 of an image and a lo-res version 12 of the same image. The hi-res version 10 is n pixels wide by t pixels high and thus has n x t pixels $P_{0,0}$ - $P_{t,n}$. But if a display device (not shown) has a

25 maximum display resolution of [n x g] pixels wide by [t x h] pixels high where g and h are less than one, then, for display purposes, one typically converts the hi-res version 10 into the lo-res version 12, which has a resolution that is less than or equal to the maximum display resolution. Therefore, to display the image on the display device with the highest possible resolution, the lo-res version 12 has (n x g) x (t x h)

30 pixels $P_{0,0}$ − $P_{(t \times h),(n \times g)}$. For example, suppose that the hi-res version 10 is n = 1920 pixels wide by t = 1088 pixels high. Furthermore, assume that the display device has a maximum resolution of n x g = 720 pixels wide by t x h = 544 pixels high. Therefore, the lo-res version 12 has a maximum horizontal resolution that is g = 3/8

of the horizontal resolution of the hi-res version 10 and has a vertical resolution that is $h = \frac{1}{2}$ of the vertical resolution of the hi-res version 10.

Referring to Figure 2, many versions of images such as the version 10 of Figure 1 are encoded using a conventional block-based compression scheme before

5   they are transmitted or stored. Therefore, for these image versions, the resolution reduction discussed above in conjunction with Figure 1 is often carried out on a block-by-block basis. Specifically, Figure 2 illustrates the down-converting example discussed above in conjunction with Figure 1 on a block level for $g = 3/8$ and $h = \frac{1}{2}$. An image block 14 of the hi-res version 10 (Figure 1) is 8 pixels wide by 8 pixels

10   high, and an image block 16 of the lo-res version 12 (Figure 1) is $8 \times 3/8 = 3$ pixels wide by $8 \times 1/2 = 4$ pixels high. The pixels in the block 16 are often called sub-sampled pixels and are evenly spaced apart inside the block 16 and across the boundaries of adjacent blocks (not shown) of the lo-res version 12. For example, referring to the block 16, the sub-sampled pixel $P_{0,2}$ is the same distance from $P_{0,1}$

15   as it is from the pixel $P_{0,0}$ in the block (not shown) immediately to the right of the block 16. Likewise, $P_{3,0}$ is the same distance from $P_{2,0}$ as it is from the pixel $P_{0,0}$ in the block (not shown) immediately to the bottom of the block 16.

Unfortunately, because the algorithms for decoding an encoded hi-res version of an image into a decoded lo-res version of the image are inefficient, an image

20   processing circuit that executes these algorithms often requires a relatively high-powered processor and a large memory and is thus often relatively expensive.

For example, U.S. Patent No. 5,262,854 describes an algorithm that decodes the encoded hi-res version of the image at its full resolution and then down-converts the decoded hi-res version into the decoded lo-res version. Therefore, because only

25   the decoded lo-res version will be displayed, generating the decoded hi-res version of the image is an unnecessary and wasteful step.

Furthermore, for encoded video images that are decoded and down converted as discussed above, the motion-compensation algorithms are often inefficient, and this inefficiency further increases the processing power and memory requirements,

30   and thus the cost, of the image processing circuit. For example, U.S. Patent 5,262,854 describes the following technique. First, a lo-res version of a reference frame is conventionally generated from a hi-res version of the reference frame and is stored in a reference-frame buffer. Next, an encoded hi-res version of a motion-

compensated frame having a motion vector that points to a macro block of the reference frame is decoded at its full resolution. But the motion vector, which was generated with respect to the hi-res version of the reference frame, is incompatible with the lo-res version of the reference frame. Therefore, a processing circuit up-

5   converts the pointed-to macro block of the lo-res version of the reference frame into a hi-res macro block that is compatible with the motion vector. The processing circuit uses interpolation to perform this up conversion. Next, the processing circuit combines the residuals and the hi-res reference macro block to generate the decoded macro block of the motion-compensated frame. Then, after the entire

10   motion-compensated frame has been decoded into a decoded hi-res version of the motion-compensated frame, the processing circuit down-converts the decoded hi-res version into a decoded lo-res version. Therefore, because reference macro blocks are down-converted for storage and display and then up-converted for motion compensation, this technique is very inefficient.

15   Unfortunately, the image processing circuits that execute the above-described down-conversion and motion-compensation techniques may be too expensive for many consumer applications. For example, with the advent of high-definition television (HDTV), it is estimated that many consumers cannot afford to replace their standard television sets with HDTV receiver/displays. Therefore, a large consumer

20   market is anticipated for HDTV decoders that down-convert HDTV video frames to standard-resolution video frames for display on standard television sets. But if these decoders incorporate the relatively expensive image processing circuits described above, then many consumers that cannot afford a HDTV receiver may also be unable to afford a HDTV decoder.

25.

Overview of Conventional Image-Compression Techniques

To help the reader more easily understand the concepts discussed above and discussed below in the description of the invention, following is a basic overview of conventional image-compression techniques.

30   To electronically transmit a relatively high-resolution image over a relatively low-band-width channel, or to electronically store such an image in a relatively small memory space, it is often necessary to compress the digital data that represents the image. Such image compression typically involves reducing the number of data bits

necessary to represent an image. For example, High-Definition-Television (HDTV) video images are compressed to allow their transmission over existing television channels. Without compression, HDTV video images would require transmission channels having bandwidths much greater than the bandwidths of existing television

5 channels. Furthermore, to reduce data traffic and transmission time to acceptable levels, an image may be compressed before being sent over the internet. Or, to increase the image-storage capacity of a CD-ROM or server, an image may be compressed before being stored thereon.

Referring to Figures 3A - 9, the basics of the popular block-based Moving
10 Pictures Experts Group (MPEG) compression standards, which include MPEG-1 and MPEG-2, are discussed. For purposes of illustration, the discussion is based on using an MPEG 4:2:0 format to compress video images represented in a Y, C_B, C_R color space. However, the discussed concepts also apply to other MPEG formats, to images that are represented in other color spaces, and to other block-based
15 compression standards such as the Joint Photographic Experts Group (JPEG) standard, which is often used to compress still images. Furthermore, although many details of the MPEG standards and the Y, C_B, C_R color space are omitted for brevity, these details are well-known and are disclosed in a large number of available references.

20 Still referring to Figures 3A - 9, the MPEG standards are often used to compress temporal sequences of images — video frames for purposes of this discussion — such as found in a television broadcast. Each video frame is divided into subregions called macro blocks, which each include one or more pixels. Figure 3A is a 16-pixel-by-16-pixel macro block 30 having 256 pixels 32 (not drawn to
25 scale). In the MPEG standards, a macro block is always 16x16 pixels, although other compression standards may use macro blocks having other dimensions. In the original video frame, i.e., the frame before compression, each pixel 32 has a respective luminance value Y and a respective pair of color-, i.e., chroma-, difference values C_B and C_R.

30 Referring to Figures 3A - 3D, before compression of the frame, the digital luminance (Y) and chroma-difference (C_B and C_R) values that will be used for compression, i.e., the pre-compression values, are generated from the original Y, C_B, and C_R values of the original frame. In the MPEG 4:2:0 format, the pre-compression

4

Y values are the same as the original Y values. Thus, each pixel 32 merely retains its original luminance value Y. But to reduce the amount of data to be compressed, the MPEG 4:2:0 format allows only one pre-compression C_B value and one pre-compression C_R value for each group 34 of four pixels 32. Each of these pre-

5 compression C_B and C_R values are respectively derived from the original C_B and C_R values of the four pixels 32 in the respective group 34. For example, a pre-compression C_B value may equal the average of the original C_B values of the four pixels 32 in the respective group 34. Thus, referring to Figures 3B - 3D, the pre-compression Y, C_B, and C_R values generated for the macro block 10 are arranged as

10 one 16 x 16 matrix 36 of pre-compression Y values (equal to the original Y values for each respective pixel 32), one 8 x 8 matrix 38 of pre-compression C_B values (equal to one derived C_B value for each group 34 of four pixels 32), and one 8 x 8 matrix 40 of pre-compression C_R values (equal to one derived C_R value for each group 34 of four pixels 32). The matrices 36, 38, and 40 are often called "blocks" of values.

15 Furthermore, because it is convenient to perform the compression transforms on 8 x 8 blocks of pixel values instead of on 16 x 16 blocks, the block 36 of pre-compression Y values is subdivided into four 8 x 8 blocks 42a – 42d, which respectively correspond to the 8 x 8 blocks A - D of pixels in the macro block 30. Thus, referring to Figures 3A – 3D, six 8 x 8 blocks of pre-compression pixel data are

20 generated for each macro block 30: four 8 x 8 blocks 42a – 42d of pre-compression Y values, one 8 x 8 block 38 of pre-compression C_B values, and one 8 x 8 block 40 of pre-compression C_R values.

Figure 4 is a block diagram of an MPEG compressor 50, which is more commonly called an encoder. Generally, the encoder 50 converts the pre-

25 compression data for a frame or sequence of frames into encoded data that represent the same frame or frames with significantly fewer data bits than the pre-compression data. To perform this conversion, the encoder 50 reduces or eliminates redundancies in the pre-compression data and reformats the remaining data using efficient transform and coding techniques.

30 More specifically, the encoder 50 includes a frame-reorder buffer 52, which receives the pre-compression data for a sequence of one or more frames and reorders the frames in an appropriate sequence for encoding. Thus, the reordered sequence is often different than the sequence in which the frames are generated and

will be displayed. The encoder 50 assigns each of the stored frames to a respective group, called a Group Of Pictures (GOP), and labels each frame as either an intra (I) frame or a non-intra (non-I) frame. For example, each GOP may include three I frames and twelve non-I frames for a total of fifteen frames. The encoder 50 always

5 encodes an I frame without reference to another frame, but can and often does encode a non-I frame with reference to one or more of the other frames in the GOP. The encoder 50 does not, however, encode a non-I frame with reference to a frame in a different GOP.

Referring to Figures 4 and 5, during the encoding of an I frame, the 8 x 8

10 blocks (Figures 3B – 3D) of the pre-compression Y, C_B, and C_R values that represent the I frame pass through a summer 54 to a Discrete Cosine Transformer (DCT) 56, which transforms these blocks of values into respective 8 x 8 blocks of one DC (zero frequency) transform value and sixty-three AC (non-zero frequency) transform values. Figure 5 is a block 57 of luminance transform values $Y\text{-}DCT_{(0, 0)a}$ - $Y\text{-}DCT_{(7,}$

15 $_{7)a}$, which correspond to the pre-compression luminance pixel values $Y_{(0, 0)a}$ - $Y_{(7, 7)a}$ in the block 36 of Figure 3B. Thus, the block 57 has the same number of luminance transform values $Y\text{-}DCT$ as the block 36 has of luminance pixel values Y. Likewise, blocks of chroma transform values $C_B\text{-}DCT$ and $C_R\text{-}DCT$ (not shown) correspond to the chroma pixel values in the blocks 38 and 40. Furthermore, the pre-compression

20 Y, C_B, and C_R values pass through the summer 54 without being summed with any other values because the summer 54 is not needed when the encoder 50 encodes an I frame. As discussed below, however, the summer 54 is often needed when the encoder 50 encodes a non-I frame.

Referring to Figure 4 and 6, a quantizer and zigzag scanner 58 limits each of

25 the transform values from the DCT 56 to a respective maximum value, and provides the quantized AC and DC transform values on respective paths 60 and 62. Figure 6 is an example of a zigzag scan pattern 63, which the quantizer and zigzag scanner 58 may implement. Specifically, the quantizer and scanner 58 reads the transform values in the transform block (such as the transform block 57 of Figure 5) in the

30 order indicated. Thus, the quantizer and scanner 58 reads the transform value in the "0" position first, the transform value in the "1" position second, the transform value in the "2" position third, and so on until it reads the transform value in the "63" position last. The quantizer and zigzag scanner 58 reads the transform values in this zigzag

pattern to increase the coding efficiency as is known. Of course, depending upon the coding technique and the type of images being encoded, the quantizer and zigzag scanner 58 may implement other scan patterns too.

Referring again to Figure 4, a prediction encoder 64 predictively encodes the

5   DC transform values, and a variable-length coder 66 converts the quantized AC transform values and the quantized and predictively encoded DC transform values into variable-length codes such as Huffman codes. These codes form the encoded data that represent the pixel values of the encoded I frame. A transmit buffer 68 then temporarily stores these codes to allow synchronized transmission of the

10 encoded data to a decoder (discussed below in conjunction with Figure 8). Alternatively, if the encoded data is to be stored instead of transmitted, the coder 66 may provide the variable-length codes directly to a storage medium such as a CD-ROM.

If the I frame will be used as a reference (as it often will be) for one or more

15 non-I frames in the GOP, then, for the following reasons, the encoder 50 generates a corresponding reference frame by decoding the encoded I frame with a decoding technique that is similar or identical to the decoding technique used by the decoder (Figure 8). When decoding non-I frames that are referenced to the I frame, the decoder has no option but to use the decoded I frame as a reference frame.

20 Because MPEG encoding and decoding are lossy — some information is lost due to quantization of the AC and DC transform values — the pixel values of the decoded I frame will often be different than the pre-compression pixel values of the original I frame. Therefore, using the pre-compression I frame as a reference frame during encoding may cause additional artifacts in the decoded non-I frame because the

25 reference frame used for decoding (decoded I frame) would be different than the reference frame used for encoding (pre-compression I frame).

Therefore, to generate a reference frame for the encoder that will be similar to or the same as the reference frame for the decoder, the encoder 50 includes a dequantizer and inverse zigzag scanner 70, and an inverse DCT 72, which are

30 designed to mimic the dequantizer and scanner and the inverse DCT of the decoder (Figure 8). The dequantizer and inverse scanner 70 first implements an inverse of the zigzag scan path implemented by the quantizer 58 such that the DCT values are properly located within respective decoded transform blocks. Next, the dequantizer

and inverse scanner 70 dequantizes the quantized DCT values, and the inverse DCT 72 transforms these dequantized DCT values into corresponding 8 x 8 blocks of decoded Y, C_B, and C_R pixel values, which together compose the reference frame. Because of the losses incurred during quantization, however, some or all of these

5 decoded pixel values may be different than their corresponding pre-compression pixel values, and thus the reference frame may be different than its corresponding pre-compression frame as discussed above. The decoded pixel values then pass through a summer 74 (used when generating a reference frame from a non-I frame as discussed below) to a reference-frame buffer 76, which stores the reference

10 frame.

During the encoding of a non-I frame, the encoder 50 initially encodes each macro-block of the non-I frame in at least two ways: in the manner discussed above for I frames, and using motion prediction, which is discussed below. The encoder 50 then saves and transmits the resulting code having the fewest bits. This technique

15 insures that the macro blocks of the non-I frames are encoded using the fewest bits.

With respect to motion prediction, an object in a frame exhibits motion if its relative position changes in the preceding or succeeding frames. For example, a horse exhibits relative motion if it gallops across the screen. Or, if the camera follows the horse, then the background exhibits relative motion with respect to the

20 horse. Generally, each of the succeeding frames in which the object appears contains at least some of the same macro blocks of pixels as the preceding frames. But such matching macro blocks in a succeeding frame often occupy respective frame locations that are different than the respective frame locations they occupy in the preceding frames. Alternatively, a macro block that includes a portion of a

25 stationary object (e.g., tree) or background scene (e.g., sky) may occupy the same frame location in each of a succession of frames, and thus exhibit "zero motion". In either case, instead of encoding each frame independently, it often takes fewer data bits to tell the decoder "the macro blocks R and Z of frame 1 (non-I frame) are the same as the macro blocks that are in the locations S and T, respectively, of frame 0

30 (reference frame)." This "statement" is encoded as a motion vector. For a relatively fast moving object, the location values of the motion vectors are relatively large. Conversely, for a stationary or relatively slow-moving object or background scene, the location values of the motion vectors are relatively small or equal to zero.

Figure 7 illustrates the concept of motion vectors with reference to the non-I frame 1 and the reference frame 0 discussed above. A motion vector MV_R indicates that a match for the macro block in the location R of frame 1 can be found in the location S of a reference frame 0. MV_R has three components. The first component,

5   here 0, indicates the frame (here frame 0) in which the matching macro block can be found. The next two components, X_R and Y_R, together comprise the two-dimensional location value that indicates where in the frame 0 the matching macro block is located. Thus, in this example, because the location S of the frame 0 has the same X-Y coordinates as the location R in the frame 1, $X_R=Y_R=0$. Conversely, the macro

10   block in the location T matches the macro block in the location Z, which has different X-Y coordinates than the location T. Therefore, X_Z and Y_Z represent the location T with respect to the location Z. For example, suppose that the location T is ten pixels to the left of (negative X direction) and seven pixels down from (negative Y direction) the location Z. Therefore, $MV_Z=(0, -10, -7)$. Although there are many other motion-

15   vector schemes available, they are all based on the same general concept. For example, the locations R may be bidirectionally encoded. That is, the location R may have two motion vectors that point to respective matching locations in different frames, one preceding and the other succeeding the frame 1. During decoding, the pixel values of these matching locations are averaged or otherwise combined to

20   calculate the pixel values of the location.

Referring again to Figure 4, motion prediction is now discussed in detail. During the encoding of a non-I frame, a motion predictor 78 compares the pre-compression Y values — the C_B and C_R values are not used during motion prediction — of the macro blocks in the non-I frame to the decoded Y values of the respective

25   macro blocks in the reference I frame and identifies matching macro blocks. For each macro block in the non-I frame for which a match is found in the I reference frame, the motion predictor 78 generates a motion vector that identifies the reference frame and the location of the matching macro block within the reference frame. Thus, as discussed below in conjunction with Figure 8, during decoding of these

30   motion-encoded macro blocks of the non-I frame, the decoder uses the motion vectors to obtain the pixel values of the motion-encoded macro blocks from the matching macro blocks in the reference frame. The prediction encoder 64 predictively encodes the motion vectors, and the coder 66 generates respective

9

codes for the encoded motion vectors and provides these codes to the transmit buffer 48.

Furthermore, because a macro block in the non-I frame and a matching macro block in the reference I frame are often similar but not identical, the encoder

5   50 encodes these differences along with the motion vector so that the decoder can account for them. More specifically, the motion predictor 78 provides the decoded Y values of the matching macro block of the reference frame to the summer 54, which effectively subtracts, on a pixel-by-pixel basis, these Y values from the pre-compression Y values of the matching macro block of the non-I frame. These

10   differences, which are called residuals, are arranged in 8 x 8 blocks and are processed by the DCT 56, the quantizer and scanner 58, the coder 66, and the buffer 68 in a manner similar to that discussed above, except that the quantized DC transform values of the residual blocks are coupled directly to the coder 66 via the line 60, and thus are not predictively encoded by the prediction encoder 44.

15   In addition, it is possible to use a non-I frame as a reference frame. When a non-I frame will be used as a reference frame, the quantized residuals from the quantizer and zigzag scanner 58 are respectively dequantized, reordered, and inverse transformed by the dequantizer and inverse scanner 70 and the inverse DCT 72, respectively, so that this non-I reference frame will be the same as the one used

20   by the decoder for the reasons discussed above. The motion predictor 78 provides to the summer 74 the decoded Y values of the reference frame from which the residuals were generated. The summer 74 adds the respective residuals from the inverse DCT 72 to these decoded Y values of the reference frame to generate the respective Y values of the non-I reference frame. The reference-frame buffer 76

25   then stores the reference non-I frame along with the reference I frame for use in motion encoding subsequent non-I frames.

Although the circuits 58 and 70 are described as performing the zigzag and inverse zigzag scans, respectively, in other embodiments, another circuit may perform the zigzag scan and the inverse zigzag scan may be omitted. For example,

30   the coder 66 can perform the zigzag scan and the circuit 58 can perform the quantization only. Because the zigzag scan is outside of the reference-frame loop, the dequantizer 70 can omit the inverse zigzag scan. This saves processing power and processing time.

10

Still referring to Figure 4, the encoder 50 also includes a rate controller 80 to insure that the transmit buffer 68, which typically transmits the encoded frame data at a fixed rate, never overflows or empties, i.e., underflows. If either of these conditions occurs, errors may be introduced into the encoded data stream. For

5   example, if the buffer 68 overflows, data from the coder 66 is lost. Thus, the rate controller 80 uses feed back to adjust the quantization scaling factors used by the quantizer/scanner 58 based on the degree of fullness of the transmit buffer 68. Specifically, the fuller the buffer 68, the larger the controller 80 makes the scale factors, and the fewer data bits the coder 66 generates. Conversely, the more empty

10   the buffer 68, the smaller the controller 80 makes the scale factors, and the more data bits the coder 66 generates. This continuous adjustment insures that the buffer 68 neither overflows or underflows.

Figure 8 is a block diagram of a conventional MPEG decompresser 82, which is commonly called a decoder and which can decode frames that are encoded by the

15   encoder 60 of Figure 4.

Referring to Figures 8 and 9, for I frames and macro blocks of non-I frames that are not motion predicted, a variable-length decoder 84 decodes the variable-length codes received from the encoder 50. A prediction decoder 86 decodes the predictively decoded DC transform values, and a dequantizer and inverse zigzag

20   scanner 87, which is similar or identical to the dequantizer and inverse zigzag scanner 70 of Figure 4, dequantizes and rearranges the decoded AC and DC transform values. Alternatively, another circuit such as the decoder 84 can perform the inverse zigzag scan. An inverse DCT 88, which is similar or identical to the inverse DCT 72 of Figure 4, transforms the dequantized transform values into pixel

25   values. For example, Figure 9 is a block 89 of luminance inverse-transform values Y-IDCT, i.e., decoded luminance pixel values, which respectively correspond to the luminance transform values Y-DCT in the block 57 of Figure 5 and to the pre-compression luminance pixel values Y_a of the block 42a of Figure 3B. But because of losses due to the quantization and dequantization respectively implemented by

30   the encoder 50 (Figure 4) and the decoder 82, the decoded pixel values in the block 89 are often different than the respective pixel values in the block 42a.

Still referring to Figure 8, the decoded pixel values from the inverse DCT 88 pass through a summer 90 — which is used during the decoding of motion-predicted

macro blocks of non-I frames as discussed below — into a frame-reorder buffer 92, which stores the decoded frames and arranges them in a proper order for display on a video display unit 94. If a decoded frame is used as a reference frame, it is also stored in the reference-frame buffer 96.

5   For motion-predicted macro blocks of non-I frames, the decoder 84, dequantizer and inverse scanner 87, and inverse DCT 88 process the residual transform values as discussed above for the transform values of the I frames. The prediction decoder 86 decodes the motion vectors, and a motion interpolator 98 provides to the summer 90 the pixel values from the reference-frame macro blocks

10   to which the motion vectors point. The summer 90 adds these reference pixel values to the residual pixel values to generate the pixel values of the decoded macro blocks, and provides these decoded pixel values to the frame-reorder buffer 92. If the encoder 50 (Figure 4) uses a decoded non-I frame as a reference frame, then this decoded non-I frame is stored in the reference-frame buffer 96.

15   Referring to Figures 4 and 8, although described as including multiple functional circuit blocks, the encoder 50 and the decoder 82 may be implemented in hardware, software, or a combination of both. For example, the encoder 50 and the decoder 82 are often implemented by a respective one or more processors that perform the respective functions of the circuit blocks.

20   More detailed discussions of the MPEG encoder 50 and the MPEG decoder 82 of Figures 4 and 8, respectively, and of the MPEG standard in general are available in many publications including "Video Compression" by Peter D. Symes, McGraw-Hill, 1998, which is incorporated by reference. Furthermore, there are other well-known block-based compression techniques for encoding and decoding both

25   video and still images.

SUMMARY OF THE INVENTION

In one aspect of the invention, an image processing circuit includes a processor that receives an encoded portion of a first version of an image. The

30   processor decodes this encoded portion directly into a decoded portion of a second version of the image, the second version having a resolution that is different than the resolution of the first version.

Therefore, such an image processing circuit can decode an encoded hi-res version of an image directly into a decoded lo-res version of the image. That is, such a circuit eliminates the inefficient step of decoding the encoded hi-res version at full resolution before down converting to the lo-res version. Thus, such an image

5   processing circuit is often faster, less complex, and less expensive than prior-art circuits that decode and down-convert images.

In another aspect of the invention, an image processing circuit includes a processor that modifies a motion vector associated with a portion of a first version of a first image. The processor then identifies a portion of a second image to which the

10   modified motion vector points, the second image having a different resolution than the first version of the first image. Next, the processor generates a portion of a second version of the first image from the identified portion of the second image, the second version of the first image having the same resolution as the second image.

Thus, such an image processing circuit can decode a motion-predicted macro

15   block using a version of a reference frame that has a different resolution than the version of the reference frame used to encode the macro block. Thus, such an image processing circuit is often faster, less complex, and less expensive than prior-art circuits that down-convert motion-predicted images.

20   BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 are pixel diagrams of a hi-res version and a lo-res version of an image.

Figure 2 are pixel diagrams of macro blocks from the hi-res and lo-res image versions, respectively, of Figure 1.

25   Figure 3A is a diagram of a conventional macro block of pixels in an image.

Figure 3B is a diagram of a conventional block of pre-compression luminance values that respectively correspond to the pixels in the macro block of Figure 3A.

Figures 3C and 3D are diagrams of conventional blocks of pre-compression chroma values that respectively correspond to the pixel groups in the macro block of

30   Figure 3A.

Figure 4 is a block diagram of a conventional MPEG encoder.

13

Figure 5 is a block of luminance transform values that are generated by the encoder of Figure 4 and that respectively correspond to the pre-compression luminance pixel values of Figure 3B.

Figure 6 is a conventional zigzag sampling pattern that can be implemented by the quantizer and zigzag scanner of Figure 4.

Figure 7 illustrates the concept of conventional motion vectors.

Figure 8 is a block diagram of a conventional MPEG decoder.

Figure 9 is a block of inverse transform values that are generated by the decoder of Figure 8 and that respectively correspond to the luminance transform values of Figure 5 and the pre-compression luminance pixel values of Figure 3B.

Figure 10 is a block diagram of an MPEG decoder according to an embodiment of the invention.

Figure 11 shows a technique for converting a hi-res, non-interlaced block of pixel values into a lo-res, non-interlaced block of pixel values according to an embodiment of the invention.

Figure 12 shows a technique for converting a hi-res, interlaced block of pixel values into a lo-res, interlaced block of pixel values according to an embodiment of the invention.

Figure 13A shows the lo-res block of Figure 11 overlaying the hi-res block of Figure 11 according to an embodiment of the invention.

Figure 13B shows the lo-res block of Figure 11 overlaying the hi-res block of Figure 11 according to another embodiment of the invention.

Figure 14 shows the lo-res block of Figure 12 overlaying the hi-res block of Figure 12 according to an embodiment of the invention.

Figure 15A shows a subgroup of transform values used to directly down-convert the hi-res block of Figure 11 to the lo-res block of Figure 11 according to an embodiment of the invention.

Figure 15B shows a subgroup of transform values used to directly down-convert the hi-res block of Figure 12 to the lo-res block of Figure 12 according to an embodiment of the invention.

Figure 16 shows substituting a series of one-dimensional IDCT calculations for a two-dimensional IDCT calculation with respect to the subgroup of transform values in Figure 15A.

14

Figure 17 shows a motion-decoding technique according to an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

5   Figure 10 is a block diagram of an image decoder and processing circuit 110 according to an embodiment of the invention. The circuit 110 includes a landing buffer 112, which receives and stores respective hi-res versions of encoded images. A variable-length decoder 114 receives the encoded image data from the landing buffer 112 and separates the data blocks that represent the image from the control

10   data that accompanies the image data. A state controller 116 receives the control data and respectively provides on lines 118, 120, and 122 a signal that indicates whether the encoded images are interlaced or non-interlaced, a signal that indicates whether the block currently being decoded is motion predicted, and the decoded motion vectors. A transform-value select and inverse zigzag circuit 124 selects the

15   desired transform values from each of the image blocks and scans them according to a desired inverse zigzag pattern. Alternatively, another circuit such as the decoder 114 can perform the inverse zigzag scan. An inverse quantizer 126 dequantizes the selected transform values, and an inverse DCT and subsampler circuit 128 directly converts the dequantized transform values of the hi-res version of

20   an image into pixel values of a lo-res version of the same image.

For I-encoded blocks, the sub-sampled pixel values from the circuit 128 pass through a summer 130 to an image buffer 132, which that stores the decoded lo-res versions of the images.

For motion-predicted blocks, a motion-vector scaling circuit 134 scales the

25   motion vectors from the state controller 116 to the same resolution as the lo-res versions of the images stored in the buffer 132. A motion compensation circuit 136 determines the values of the pixels in the matching macro block that is stored in the buffer 136 and that is pointed to by the scaled motion vector. In response to the signal on the line 120, a switch 137 couples these pixel values from the circuit 136 to

30   the summer 130, which respectively adds them to the decoded and sub-sampled residuals from the circuit 128. The resultant sums are the pixel values of the decoded macro block, which is stored in the frame buffer 132. The frame buffer 132

stores the decoded lo-res versions of the images in display order and provides the lo-res versions to an HDTV receive/display 138.

Figure 11 illustrates the resolution reduction performed by the IDCT and sub-sampler circuit 128 of Figure 10 on non-interlaced images according to an

5   embodiment of the invention. Although the circuit 128 converts an encoded hi-res version of a non-interlaced image directly into a decoded lo-res version of the image, for clarity, Figure 11 illustrates this resolution reduction in the pixel domain. Specifically, an 8 x 8 block 140 of pixels P from the hi-res version of the image is down converted to a 4 x 3 block 142 of sub-sampled pixels S. Therefore, in this

10   example, the horizontal resolution of the block 142 is 3/8 the horizontal resolution of the block 140 and the vertical resolution of the block 142 is 1/2 the vertical resolution of the block 140. The value of the sub-sampled pixel S_{00} in the block 142 is determined from a weighted combination of the values of the pixels P in the sub-block 144 of the block 140. That is, S_{00} is a combination of $w_{00}P_{00}$, $w_{01}P_{01}$, $w_{02}P_{02}$,

15   $w_{03}P_{03}$, $w_{10}P_{10}$, $w_{11}P_{11}$, $w_{12}P_{12}$, and $w_{13}P_{13}$, where w_{00} - w_{13} are the respective weightings of the values for P_{00} - P_{13}. The calculation of the weightings w are discussed below in conjunction with Figures 13a and 13b. Likewise, the value of the sub-sampled pixel S_{01} is determined from a weighted combination of the values of the pixels P in the sub-block 146, the value of the sub-sampled pixel S_{02} is

20   determined from a weighted combination of the values of the pixels P in the sub-block 148, and so on. Furthermore, although the blocks 140 and 142 and the sub-blocks 144, 146, and 148 are shown having specific dimensions, they may have other dimensions in other embodiments of the invention.

Figure 12 illustrates the resolution reduction performed by the IDCT and sub-

25   sampler circuit 128 of Figure 10 on interlaced images according to an embodiment of the invention. Although the circuit 128 converts an encoded hi-res version of an interlaced image directly into a decoded lo-res version of the image, for clarity, Figure 12 illustrates this resolution reduction in the pixel domain. Specifically, an 8 x 8 block 150 of pixels P from the hi-res version of the image is down converted to a 4

30   x 3 block 152 of sub-sampled pixels S. Therefore, in this example, the horizontal resolution of the block 152 is 3/8 the horizontal resolution of the block 150 and the vertical resolution of the block 152 is 1/2 the vertical resolution of the block 150. The value of the sub-sampled pixel S_{00} in the block 152 is determined from a weighted

combination of the values of the pixels P in the sub-block 154 of the block 150. That is, S_{00} is a combination of $w_{00}P_{00}$, $w_{01}P_{01}$, $w_{02}P_{02}$, $w_{03}P_{03}$, $w_{20}P_{20}$, $w_{21}P_{21}$, $w_{22}P_{22}$, and $w_{23}P_{23}$, where $w_{00} - w_{23}$ are the respective weightings of the values for $P_{00} - P_{23}$. Likewise, the value of the sub-sampled pixel S_{01} is determined from a weighted

5　　combination of the values of the pixels P in the sub-block 156, the value of the sub-sampled pixel S_{02} is determined from a weighted combination of the values of the pixels P in the sub-block 158, and so on. Furthermore, although the blocks 150 and 152 and the sub-blocks 154, 156, and 158 are shown having specific dimensions, they may have other dimensions in other embodiments of the invention.

10　　　　　Figure 13A shows the lo-res block 142 of Figure 11 overlaying the hi-res block 140 of Figure 11 according to an embodiment of the invention. Block boundaries 160 are the boundaries for both of the overlaid blocks 140 and 142, the sub-sampled pixels S are marked as X's, and the pixels P are marked as dots. The sub-sampled pixels S are spaced apart by a horizontal distance Dsh and a vertical distance Dsv

15　　both within and across the block boundaries 160. Similarly, the pixels P are spaced apart by a horizontal distance Dph and a vertical distance Dpv. In the illustrated example, Dsh = 8/3(Dph) and Dsv = 2(Dpv). Because S_{00} is horizontally aligned with and thus horizontally closest to the pixels P_{01} and P_{11}, the values of these pixels are weighted more heavily in determining the value of S_{00} than are the values of the

20　　more horizontally distant pixels P_{00}, P_{10}, P_{02}, P_{12}, P_{03}, and P_{13}. Furthermore, because S_{00} is halfway between row 0 (i.e., P_{00}, P_{01}, P_{02}, and P_{03}) and row 1 (i.e., P_{10}, P_{11}, P_{12}, and P_{13}) of the pixels P, all the pixels P in rows 0 and 1 are weighted equally in the vertical direction. For example, in one embodiment, the values of the pixels P_{00}, P_{02}, P_{03}, P_{10}, P_{12}, and P_{13} are weighted with w = 0 such that they

25　　contribute nothing to the value of S_{00}, and the values P_{01} and P_{11} are averaged together to obtain the value of S_{00}. The values of S_{01} and S_{02} are calculated in a similar manner using the weighted values of the pixels P in the sub-blocks 146 and 148 (Figure 11), respectively. But because the sub-sampled pixels S_{00}, S_{01}, and S_{02} are located at different horizontal positions within their respective sub-blocks 144,

30　　146, and 148, the sets of weightings w used to calculate the values of S_{00}, S_{01}, and S_{02} are different from one another. The values of the remaining sub-sampled pixels S are calculated in a similar manner.

Figure 13B shows the lo-res block 142 of Figure 11 overlaying the hi-res block 140 of Figure 11 according to another embodiment of the invention. A major difference between the overlays of Figures 13A and 13B is that in the overlay of Figure 13B, the sub-sampled pixels S are horizontally shifted to the left with respect

5   to their positions in Figure 13A. Because of this shift, the pixel weightings w are different than those used in Figure 13A. But other than the different weightings, the values of the sub-sampled pixels S are calculated in a manner similar to that described above in conjunction with Figure 13A.

Figure 14 shows the lo-res block 152 of Figure 12 overlaying the hi-res block

10   150 of Figure 12 according to an embodiment of the invention. The sub-sampled pixels S have the same positions as in Figure 13a, so the horizontal weightings are the same as those for Figure 13a. But because the pixels P and sub-sampled pixels S are interlaced, the pixels S are not halfway between row 0 (i.e., P_{00}, P_{01}, P_{02}, and P_{03}) and row 1 (i.e., P_{20}, P_{21}, P_{22}, and P_{23}) of the sub-block 154. Therefore, the pixels

15   P in row 0 are weighted more heavily than the respective pixels P in row 1. For example, in one embodiment, the values of the pixels P_{00}, P_{02}, P_{03}, P_{20}, P_{22}, and P_{23} are weighted with w = 0 such that they contribute nothing to the value of S_{00}, and the value of P_{01} is weighted more heavily that the value of P_{21}. For example, the value of S_{00} can be calculated by straight-line interpolation, i.e., bilinear filtering, between the

20   values of P_{01} and P_{21}.

The techniques described above in conjunction with Figures 13A, 13B, and 14 can be used to calculate the luminance or chroma values of the sub-sampled pixels S.

Referring to Figures 10 and 15A, the variable length decoder 114 provides a

25   block 160 of transform values (shown as dots), which represent a block of an encoded, non-interlaced image, to the selection and inverse zigzag circuit 124. The circuit 124 selects and uses only a sub-block 162 of the transform values to generate the values of the non-interlaced sub-sampled pixels S of Figures 11, 13A, and 13B. Because the circuit 110 decodes and down-converts the received images to a lower

30   resolution, the inventors have found that much of the encoded information, i.e., many of the transform values, can be eliminated before the inverse DCT and sub-sampler circuit 128 decodes and down-converts the encoded macro blocks. Eliminating this information significantly reduces the processing power and time that the decoder 110

●   ···   —   ●

requires to decode and down-convert encoded images. Specifically, the lo-res version of the image lacks the fine detail of the hi-res version, and the fine detail of an image block is represented by the higher-frequency transform values in the corresponding transform block. These higher-frequency transform values are

5   located toward and in the lower right-hand quadrant of the transform block. Conversely, the lower-frequency transform values are located toward and in the upper left-hand quadrant, which is equivalent to the sub-block 162. Therefore, by using the sixteen lower-frequency transform values in the sub-block 162 and discarding the remaining forty eight higher-frequency transform values in the block

10   160, the circuit 128 does not waste processing power or time incorporating the higher-frequency transform values into the decoding and down-converting algorithms. Because these discarded higher-frequency transform values would make little or no contribution to the decoded lo-res version of the image, discarding these transform values has little or no effect on the quality of the lo-res version.

15   Figure 15A is a block 164 of transform values that represent an encoded, interlaced image, and a sub-block 166 of the transform values that the circuit 124 uses to generate the values of the interlaced sub-sampled pixels S of Figures 12 and 14. The inventors found that the transform values in the sub-block 166 give good decoding and down-converting results. Because the sub-block 166 is not in matrix

20   form, the inverse zigzag scan pattern of the circuit 124 can be modified such that the circuit 124 scans the transform values from the sub-block 166 into a matrix form such as a 4 x 4 matrix.

Referring to Figures 10 – 15B, the mathematical details of the decoding and sub-sampling algorithms executed by the decoder 110 are discussed. For example

25   purposes, these algorithms are discussed operating on a sub-block of the non-interlaced block 57 of luminance values Y (Figure 5), where the sub-block is the same as the sub-block 162 of Figure 15B.

For an 8 x 8 block of transform values f(u,v), the inverse DCT (IDCT) transform is:

30

1) $$F(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v f(u,v) \cos\left[\frac{2(x+1)u\pi}{16}\right] \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

where F(x,y) is the IDCT value, *i.e.*, the pixel value, at the location x, y of the 8 x 8 IDCT matrix. The constants C_u and C_v are known, and their specific values are not important for this discussion. Equation 1 can be written in matrix form as:

5

$$2) \qquad P(x,y) = \begin{bmatrix} Y_{DCT00} \cdots Y_{DCT07} \\ \vdots \quad \vdots \\ Y_{DCT70} \cdots Y_{DCT77} \end{bmatrix} \bullet \begin{bmatrix} D_{(x,y)00} \cdots D_{(x,y)07} \\ \vdots \quad \vdots \\ D_{(x,y)70} \cdots D_{(x,y)77} \end{bmatrix}$$

where P(x, y) is the pixel value being calculated, the matrix Y_{DCT} is the matrix of transform values $Y_{DCT(u,v)}$ for the corresponding block decoded pixel values to which P(x,y) belongs, and the matrix D(x,y) is the matrix of constant coefficients that

10   represent the values on the left side of equation (1) other than the transform values f(u,v). Therefore, as equation (2) is solved for each pixel value P(x, y), Y_{DCT} remains the same, and D(x, y), which is a function of x and y, is different for each pixel value P being calculated.

The one-dimensional IDCT algorithm is represented as follows:

15

$$3) \qquad F(x) = \frac{1}{2} \sum_{u=0}^{7} C_u f(u) \cos\left[\frac{(2x+1)u\pi}{16} \right]$$

where F(x) is a single row of inverse transform values, and f(u) is a single row of transform values. In matrix form, equation (3) can be written as:

20

$$4) \qquad [P_0 \cdots P_7] = [Y_{DCT0} \cdots Y_{DCT7}] \bullet \begin{bmatrix} D_{00} \cdots D_{07} \\ \vdots \quad \vdots \\ D_{70} \cdots D_{77} \end{bmatrix}$$

where each of the decode pixel values P equals the inner product of the row of transform values $Y_{DCT0} - Y_{DCT7}$ with each respective row of the matrix **D**. That is, for

25   example $P_0 = [Y_{DCT0}, \ldots, Y_{DCT7}] \cdot [D_{00}, \ldots, D_{07}]$, and so on. Thus, more generally in the one-dimensional case, a pixel value P_X can be derived according to the following equation:

$$5) \qquad P_i = Y_{DCT} \bullet D_i$$

where D_i is the ith row of the matrix D of equation (4). Now, as stated above in conjunction with Figure 11, the values of a number of pixels in the first and second rows of the sub-block 144 are combined to generate the sub-sampled pixel S_{00}.

5   However, for the moment, let's assume that only row 0 of the pixels P exists, and that only one row of sub-sampled pixels S_0, S_1, and S_2 is to be calculated. Applying the one-dimensional IDCT of equations (4) and (5) to a single row such as row 0, we get the following equation:

10   6) $$S_z = \sum_{i=0}^{n} W_i \bullet P_i$$

Where S_z is the value of the sub-sampled pixel, W_i is the weighting factor for a value of a pixel P_i, and $i = 0 - n$ represents the locators of the particular pixels P within the row that contribute to the value of S_z. For example, still assuming that only

15   row 0 of the pixels P is present in the sub-block 144, we get the following for S_0:

7) $$S_0 = \sum_{i=0}^{3} W_i \bullet P_i$$

where P_i equals the values of P_0, P_1, P_2, and P_3 for $i = 0 - 3$. Now, using equation (5)

20   to substitute for P, we get the following:

8) $$S_z = \sum_{i=0}^{n} W_i \bullet D_i \bullet Y_{DCT} = \left(\sum_{c=0}^{n} w_i \bullet D_i \right) \bullet Y_{DCT} = Rz \bullet Y_{DCT}$$

where r_z = the sum of w_i . D_i for $i = 0 - n$. Therefore, we have derived a one-

25   dimensional equation that relates the sub-sampled pixel value S_z directly to the corresponding one-dimensional matrix Y_{DCT} of transform values and the respective rows of the coefficients D_i. That is, this equation allows one to calculate the value of S_z without having to first calculate the values of P_i.

Now, referring to the two-dimensional equations (1) and (2), equation (5) can

30   be extended to two dimensions as follows:

21

9) $P_{x,y} = D_{x,y} * Y_{DCT} = D_{x,y(0,0)} \bullet Y_{DCT(0,0)} \cdots + D_{x,y(7,7)} \bullet Y_{DCT(7,7)}$

where the asterisk indicates an inner product between the matrices. The inner
product means that every element of the matrix $D_{X,Y}$ is multiplied by the respective
element of the matrix Y_{DCT}, and the sum of these products equals the value of $P_{X,Y}$.
Equation (8) can also be converted into two dimensions as follows:

10) $S_{yz} = \left(\sum_{i=0}^{n} w_i \bullet D_i \right) * Y_{DCT} = R_{yz} * Y_{DCT}$

Therefore, the matrix R_{YZ} is a sum of the weighted matrices D_i from $I = 0 - n$. For
example, referring again to FIG. 11, the value of the sub-sampled pixel S_{00} is given
by:

11) $S_{00} = \left(\sum_{i=0}^{7} w_i \bullet D_i \right) * Y_{DCT} = R_{00} * Y_{DCT}$

where i = 0 to 7 corresponds to the values of P_{00}, P_{01}, P_{02}, P_{03}, P_{10}, P_{11}, P_{12}, and P_{13},
respectively. Thus, the circuit 124 of Figure 10 calculates the value of the sub-
sampled pixel S_{00} directly from the transform values and the associated transform
coefficient matrices. Therefore, the circuit 124 need not perform an intermediate
conversion into the pixel values P.

Equation (11) is further simplified because as stated above in conjunction with
Figure 15A, only sixteen transform values in the sub-block 162 are used in equation
(11).. Therefore, since we are doing an inner product, the matrix R_{YZ} need only have
sixteen elements that correspond to the sixteen transform values in the sub-block
162. This reduces the number of calculations and the processing time by
approximately one fourth.

Because in the above example there are sixteen elements in both the
matrices R_{YZ} and Y_{DCT}, a processor can arrange each of these matrices as a single-
dimension matrix with sixteen elements to do the inner product calculation.

Alternatively, if the processing circuit works more efficiently with one-dimensional vectors each having four elements, both matrices R_{YZ} and Y_{DCT} can be arranged into four respective one-dimensional, four element vectors, and thus the value of a sub-sampled pixel S_{YZ} can be calculated using four inner-product calculations. As stated

5   above in conjunction with Figure 15B, for an interlaced image or for any transform-value sub-block that does not initially yield an efficient matrix, the inverse zigzag scanning algorithm of the block 124 of Figure 10 can be altered to place the selected transform values in an efficient matrix format.

Referring to Figure 16, in another embodiment of the invention, the values of

10   the sub-sampled pixels S_{YZ} are calculated using a series of one-dimensional IDCT calculations instead of a single two-dimensional calculation. Specifically, Figure 16 illustrates performing such a series of one-dimensional IDCT calculations for the sub-block 162 of transform values. This technique, however, can be used with other sub-blocks of transform values such as the sub-block 166 of Figure 15B. Because

15   the general principles of this one-dimensional technique are well known, this technique is not discussed further.

Next, calculation of the weighting values W_i is discussed for the sub-sampling examples discussed above in conjunction with Figures 11 and 13A according to an embodiment of the invention. As discussed above in conjunction with Figure 13A,

20   because the sub-sampled pixels S_{00} - S_{02} are halfway between the first and second rows of the pixels P, the weighting values W for values of the pixels in the first row are the same as the respective weighting values W for the values of the pixels in the second row. Therefore, for the eight pixel values in the sub-block 144, we only need to calculate four weighting values W. To perform the weighting, in one embodiment

25   a four-tap (one tap for each of the four pixel values) Lagrangian interpolator with fractional delays of 1, 1-2/3, and 1-1/2, respectively for the for the sub-sampled pixel values S_{00} - S_{02}. In one embodiment, the weighting values w are assigned according to the following equations:

30   12) $W_0 = -\frac{1}{6}(d-1)(d-2)(d-3)$

           13) $W_1 = \frac{1}{2}(d)(d-2)(d-3)$

14) $W_2 = -\frac{1}{2}(d)(d-1)(d-3)$

15) $W_3 = \frac{1}{6}(d)(d-1)(d-2)$

5

Referring to Figure 13A, the first two delays 1 and 1-2/3, correspond to the sub-sampled pixel values S_{00} and S_{01}. Specifically, the delays indicate the positions of the sub-sampled pixels S_{00} and S_{01} with respect to the first, *i.e.*, leftmost, pixel P in the respective sub-groups 144 and 146 (Figure 11) of pixels P. For example,

10   because S_{00} is aligned with P_{01} and P_{11}, it is one pixel-separation D_{ph} from the first pixels P_{00} and P_{01} in a horizontal direction. Therefore, when the delay value of 1 is plugged into the equations 12 - 15, the only weighting w with a non-zero value is w_1, which corresponds to the pixel values P_{01} and P_{11}. This makes sense because the pixel S_{00} is aligned directly with P_{01} and P_{11}, and, therefore, the weighting values for

15   the other pixels P can be set to zero. Likewise, referring to Figures 11 and 13A, the sub-sampled pixel S_1 is 1-2/3 pixel-separations D_{ph} from the first pixels P_{02} and P_{12} in the sub-block 146. Therefore, because the pixel S_{01} is not aligned with any of the pixels P, then none of the weighting values w equals zero. Thus, for the sub-sampled pixel S_{01}, W_0 is the weighting value for the values of P_{02} and P_{12}, W_1 is the

20   weighting value for the values of P_{03} and P_{13}, W_2 is the weighting values for the values of P_{04} and P_{14}, and W_3 is the weighting value for the values of P_{05} and P_{15}.

In one embodiment, the delay for the sub-sampled pixel S_{02} is calculated differently than for the sub-sampled pixels S_{00} and S_{01}. To make the design of the Lagrangian filter more optimal, it is preferred to use a delay for S_{02} of 1-1/3.

25   Conversely, if the delay is calculated in the same way as the delays for S_{00} and S_{01}, then it would follow that because S_{02} is 2-1/3 pixel-separations D_{ph} from the first pixel P_{04} in the sub-group 148, the delay should be 2-1/3. However, so that the optimal delay of 1-1/3 can be used, we calculate the delay as if the pixels P_{05} and P_{15} are the first pixels in the sub-group 148, and then add two fictional pixels P_{08} and P_{18}, which

30   are given the same values as P_{07} and P_{17}, respectively. Therefore, the weighting functions w_0 - w_3 correspond to the pixels P_{05}, P_{15}, P_{06} and P_{16}, P_{07} and P_{17}, and the fictitious pixels P_{08} and P_{18}, respectively. Although this technique for calculating the

24

delay for S_{02} may not be as accurate as if we used a delay of 2-1/3, the increase in the Lagrangian filter's efficiency caused by using a delay of 1-1/3 makes up for this potential inaccuracy.

5 Furthermore, as stated above, because all of the sub-sampled pixels S_{00} - S_{02} lie halfway between the rows 0 and 1 of pixels P, a factor of 1/2 can be included in each of the weighting values so as to effectively average the weighted values of the pixels P in row 0 with the weighted values of the pixels P in the row 1. Of course, if the sub-sampled pixels S_{00} - S_{02} were not located halfway between the rows, then a second Lagrangian filter could be implemented in a vertical direction in a manner 10 similar to that described above for the horizontal direction. Or, the horizontal and vertical Lagrangian filters could be combined into a two-dimensional Lagrangian filter.

Referring to Figures 12 and 14, for the interlaced block 150, the sub-sampled pixels S_{00} - S_{02} are vertically located one-fourth of the way down between the rows 0 15 and 2 of pixels. Therefore, in addition to being multiplied by the respective weighting values w_i, the values of the pixels P in the respective sub-blocks can be bilinearly weighted. That is, the values of the pixels in row 0 are vertically weighted by 3/4 and the values of the pixels in row 2 are vertically weighted by 1/4 to account for the uneven vertical alignment. Alternatively, if the sub-sampled pixels S from block to 20 block do not have a constant vertical alignment with respect to the pixels P, then a Lagrangian filter can be used in the vertical direction.

The above-described techniques for calculating the values of the sub-sampled pixels S can be used to calculate both the luminance and chroma values of the pixels S.

25 Referring to Figure 17, the motion compensation performed by the decoder 110 of Figure 10 is discussed according to an embodiment of the invention. For example purposes, assume that the encoded version of the image is non-interlaced and includes 8 x 8 blocks of transform values, and that the circuit 124 of Figure 10 decodes and down-coverts these encoded blocks into 4 x 3 blocks of sub-sampled 30 pixels S such as the block 142 of Figure 11. Furthermore, assume that the encoded motion vectors have a resolution of 1/2 pixel in the horizontal direction and 1/2 pixel in the vertical direction. Therefore, because the lo-res version of the image has 3/8 the horizontal resolution and 1/2 the vertical resolution of the hi-res version of the

image, the scaled motion vectors from the circuit 134 (Figure 10) have a horizontal resolution of $3/8 \times 1/2 = (3/16)D_{sh}$ and a vertical resolution of $1/2 \times 1/2 = (1/4)D_{sv}$. Thus, the horizontal fractional delays are multiples of 1/16 and the vertical fractional delays are multiples of 1/4. Also assume that the encoded motion vector had a

5   value of 2.5 in the horizontal direction and a value of 1.5 in the vertical direction. Therefore, the example scaled motion vector equals $2\text{-}1/2 \times 3/8 = 15/16$ in the horizontal direction and $1\text{-}1/2 \times 1/2 = 3/4$ in the vertical direction. Thus, this scaled motion vector points to a matching macro block 170 whose pixels S are represented by "x".

10   The pixels of the block 170, however, are not aligned with the pixels S (represented by dots) of the reference macro block 172. The reference block 172 is larger than the matching block 170 such that it encloses the area within which the block 170 can fall. For example, the pixel S_{00} can fall anywhere between or on the reference pixels S_I, S_J, S_M, and S_N. Therefore, in a manner similar to that described

15   above for the pixels S of the block 142 (Figure 11), each pixel S of the matching block 170 is calculated from the weighted values of respective pixels S in a filter block 174, which includes the blocks 170 and 172. In the illustrated embodiment, each pixel S of the block 170 is calculated from a sub-block of $4 \times 4 = 16$ pixels from the filter block 174. For example, the value of S_{00} is calculated from the weighted

20   values of the sixteen pixels S in a sub-block 176 of the filter block 174.

        In one embodiment, a four-tap polyphase Finite Impulse Response FIR filter (e.g., a Lagrangian filter) having a delay every $(1/16) D_{sh}$ is used in the horizontal direction, and a four-tap FIR filter having a delay every $(1/4) D_{sv}$ is used in the vertical direction. Therefore, one could think of the combination of these two filters

25   as a set of $16 \times 4 = 64$ two-dimensional filters for each respective phase in both the horizontal and vertical directions. In this example, the pixel S_{00} is horizontally located $(1\text{-}15/16)D_{sh}$ from the first column of pixels (i.e., S_a, S_h, S_l, and S_q) in the sub-block 176 and the horizontal contributions to the respective weighting values w are calculated in a manner similar to that discussed above in conjunction with Figure

30   13A. Likewise, the pixel S_{00} is vertically located $(1\text{-}3/4)D_{sv}$ from the first row of pixels (i.e., S_a-S_d) in the sub-block 176 and vertical contributions of the weighting functions are calculated in a manner similar to that used to calculate the horizontal contributions. The horizontal and vertical contributions are then combined to obtain

the weighting function for each pixel in the sub-block 176 with respect to S_{00}, and the value of S_{00} is calculated using these weighting functions. The values of the other pixels S in the matching block 170 are calculated in a similar manner. For example, the value of the pixel S_{01} is calculated using the weighted values of the pixels in the

5   sub-block 178, and the value of the pixel S_{10} is calculated using the values of the pixels in the sub-block 180.

Therefore, all of the motion compensation pixels S_{00} - S_{75} are calculated using 4 multiply-accumulates (MACS) x 6 pixels per row x 11 rows (in the filter block 174) = 260 total MACS for horizontal filtering, and 4 MACS x 8 pixels per column x 9

10   columns = 288 MACS for vertical filtering for a total of 552 MACS to calculate the pixel values of the matching block 170. Using a vector image processing circuit that operates on 1 x 4 vector elements, we can break down the horizontal filtering into 264 ÷ 4 = 66 1 x 4 inner products, and can break down the vertical filtering into 288 ÷ 4 = 72 1 x 4 inner products.

15   Referring to Figures 10 and 15, once the motion compensation circuit 136 calculates the values of the pixels in the matching block 170, the summer 130 adds these pixel values to the respective residuals from the inverse DCT and sub-sample circuit 128 to generate the decoded lo-res version of the image. Then, the decoded macro block is provided to the frame buffer 132 for display on the HDTV

20   receiver/display 138. If the decoded macro block is part of a reference frame, it may also be provided to the motion compensator 136 for use in decoding another motion-predicted macro block.

The motion decoding for the pixel chroma values can be performed in the same manner as described above. Alternatively, because the human eye is less

25   sensitive to color variations than to luminance variations, one can use bilinear filtering instead of the more complicated Lagrangian technique described above and still get good results.

Furthermore, as discussed above in conjunction with Figure 7, some motion-predicted macro blocks have motion vectors that respectively point to matching

30   blocks in different frames. In such a case, the values of the pixels in each of the matching blocks is calculated as described above in conjunction with Figure 16, and are then averaged together before the residuals are added to produce the decoded macro block. Alternatively, one can reduce processing time and bandwidth by using

only one of the matching blocks to decode the macro block. This has been found to produce pictures of acceptable quality with a significant reduction in decoding time.

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various
5   modifications may be made without deviating from the spirit and scope of the invention. For example, although down-conversion of an image for display on a lower-resolution display screen is discussed, the above-described techniques have other applications. For example, these techniques can be used to down-convert an image for display within another image. This is often called Picture-In-Picture (PIP)
10 display. Additionally, although the decoder 110 of Figure 10 is described as including a number of circuits, the functions of these circuits may be performed by one or more conventional or special-purpose processors or may be implemented in hardware.

What is claimed:

1.   An image processing circuit, comprising:
a processor operable to,

5   receive an encoded portion of a first version of an image, the first
version having a resolution; and
          convert the encoded portion directly into a decoded portion of a second
version of the image, the second version having a resolution that is different
than the resolution of the first version.

10

2.   The processing circuit of claim 1 wherein the resolution of the second
version of the image is lower than the resolution of the first version of the image.

3.   The processing circuit of claim 1 wherein:

15   the encoded portion of the first version of the image is represented by
transform values; and
          the decoded portion of the second version of the image is represented by
pixel values.

20   4.   An image processing circuit, comprising:
a processor operable to,
          receive a first group of transform values that represents a portion of a
first version of an image,
          select a second group of transform values from the first group, the

25   second group having fewer transform values than the first group, and
          convert the second group of transform values directly into a first group
of pixel values that represents a portion of a second version of the image, the
second version of the image having fewer pixels than the first version of the
image.

30

5.   The processing circuit of claim 4 wherein each of the transform values
in the first group comprises a respective Discrete-Cosine-Transform value.

29

6.   The processing circuit of claim 4 wherein:

the first group of transform values comprises an 8 x 8 block of transform values, the block having four quadrants; and

the second group of transform values comprises the transform values from a

5   quadrant of the block.

7.   The processing circuit of claim 4 wherein:

the first group of transform values comprises an 8 x 8 block of transform values, the block having an upper-left quadrant; and

10   the second group of transform values comprises the transform values from the upper-left quadrant of the block.

8.   The processing circuit of claim 4 wherein:

the first group of transform values comprises an 8-row x 8-column block of

15   transform values; and

the second group of transform values comprises the first three transform values from each of the first four rows of the block and the first transform value from each of the last four rows of the block.

20   9.   The processing circuit of claim 4 wherein:

the image comprises a video frame;

the portion of the first version of the video frame is non-interlaced; and

the portion of the second version of the video frame is non-interlaced.

25   10.   The processing circuit of claim 4 wherein:

the image comprises a video frame;

the portion of the first version of the video frame is interlaced; and

the portion of the second version of the video frame is interlaced.

30   11.   The processing circuit of claim 4 wherein:

the first version of the image is 1920 pixels wide by 1088 pixels high; and

the second version of the image is 720 pixels wide by 544 pixels high.

12. The processing circuit of claim 4 wherein:

the first group of transform values represents a second group of pixel values that represents the portion of the first version of the image; and

the processor is operable to convert the second group of transform values

5   directly into a pixel value of the first group of pixel values by mathematically combining transform coefficients associated with pixel values of the second group of pixel values.

13. The processing circuit of claim 4 wherein:

10   the first group of transform values represents a second group of pixel values that represents the portion of the first version of the image; and

the processor is operable to convert the second group of transform values directly into a pixel value of the first group of pixel values by,

weighting transform coefficients associated with pixel values of the

15   second group of pixel values, and

mathematically combining the weighted transform coefficients.

14. The processing circuit of claim 4 wherein:

the first group of transform values represents a second group of pixel values

20   that represents the portion of the first version of the image; and

the processor is operable to convert the second group of transform values directly into a pixel value of the first group of pixel values by,

weighting transform coefficients associated with pixel values of the second group of pixel values, and

25   summing corresponding ones of the weighted transform coefficients.

15. The processing circuit of claim 4 wherein:

each of the transform values in the first group of transform values respectively comprises a Discrete-Cosine-Transform value;

30   the first group of transform values represents a second group of pixel values that represents the portion of the first version of the image; and

the processor is operable to convert the second group of transform values directly into a pixel value of the first group of pixel values by,

weighting Inverse-Discrete-Cosine-Transform coefficients associated with pixel values of the second group of pixel values,

summing corresponding ones of the weighted coefficients, and

mathematically combining the second group of transform values and

5   the weighted coefficients according to an Inverse-Discrete-Cosine-Transform algorithm.

16.   An image processing circuit, comprising:

a processor operable to,

10   modify a motion vector associated with a portion of a first version of a first image,

identify a portion of a second image to which the modified motion vector points, the second image having a different resolution than the first version of the first image, and

15   generate a portion of a second version of the first image from the identified portion of the second image, the second version of the first image having the same resolution as the second image.

17.   The image processing circuit of claim 16 wherein the second image

20   has a lower resolution than the first version of the first image.

18.   The image processing circuit of claim 16 wherein:

the motion vector is compatible with the first version of the first image; and

the processor is operable to modify the motion vector to be compatible with

25   the second image.

19.   An image processing circuit, comprising:

a processor operable to,

modify a motion vector associated with a portion of a first version of a

30   first image to be compatible with a second image having a different resolution than the first version of the first image,

identify a portion of the second image to which the modified motion vector points,

convert a first group of residuals that represents the portion of the first version of the first image into a second group of residuals that represents a portion of a second version of the first image, the second version having the same resolution as the second image, and

5   mathematically combine the second group of residuals and pixel values that represent the identified portion of the second image to generate pixel values that represent the portion of the second version of the first image.

20.   The image processing circuit of claim 19 wherein the second image

10   and the second version of the first image have a lower resolution than the first version of the first image.

21.   The image processing circuit of claim 19 wherein:

the second image and the second version of the first image have a lower

15   resolution than the first version of the first image; and

the second group of residuals has fewer residuals than the first group of residuals.

22.   The image processing circuit of claim 19 wherein the processor is

20   operable to modify the motion vector by multiplying the motion vector with a scaling factor between the first version of the first image and the second image.

23.   The image processing circuit of claim 19 wherein the modified motion vector has a resolution of less than ½ pixel in at least one dimension.

25

24.   The image processing circuit of claim 19 wherein the processor is operable to calculate the pixel values that represent the identified portion of the second image.

30   25.   The image processing circuit of claim 19 wherein the processor is operable to mathematically combine by summing each of the residuals in the second group with a respective one of the pixel values that represent the identified portion of the second image.

26.　The image processing circuit of claim 19 wherein the pixel values that represent the identified portion of the second image correspond to interpolated pixels that are offset from actual pixels of the second image.

5

27.　The image processing circuit of claim 19 wherein the processor is operable to convert the first group of residuals by:

selecting a first group of transform values from a second group of transform values that represents the first group of residuals, the first group of transform values

10　being smaller than the second group of transform values; and

converting the first group of transform values directly into the second group of residuals.

28.　A method, comprising:

15　receiving an encoded portion of a first version of an image, the first version having a resolution; and

converting the encoded portion directly into a decoded portion of a second version of the image, the second version having a resolution that is different than the resolution of the first version.

20

29.　The method of claim 28 wherein the resolution of the first version of the image is higher than the resolution of the second version of the image.

30.　The method of claim 28 wherein:

25　the receiving comprises receiving transform values that represent the encoded portion of the first version of the image; and

the converting comprises converting the transform values into pixel values that represent the decoded portion of the second version of the image.

30　31.　A method, comprising:

receiving a first group of transform values that represents a portion of a first version of an image;

selecting a second group of transform values from the first group, the second group being smaller than the first group; and

converting the second group of transform values directly into a first group of pixel values that represents a portion of a second version of the image, the second

5 version having fewer pixels than the first version.

32. The method of claim 31 wherein:

the image comprises a video frame;

the portion of the first version of the video frame is non-interlaced;

10 the portion of the second version of the video frame is non-interlaced;

the first group of transform values comprises an 8 x 8 block of transform values, the block having an upper-left quadrant; and

the second group of transform values comprises the transform values from the upper-left quadrant of the block.

15

33. The method of claim 31 wherein:

the image comprises a video frame;

the portion of the first version of the video frame is interlaced;

the portion of the second version of the video frame is interlaced;

20 the first group of transform values comprises an 8-row x 8-column block of transform values; and

the second group of transform values comprises the first three transform values from each of the first four rows of the block and the first transform value from each of the last four rows of the block.

25

34. The method of claim 31 wherein:

the first group of transform values represents a second group of pixel values that represents the portion of the first version of the image; and

the converting comprises mathematically combining transform coefficients

30 associated with respective subgroups of pixel values from the second group of pixel values to generate each pixel value in the first group of pixel values.

35.　The method of claim 31 wherein:

the first group of transform values represents a second group of pixel values that represents the portion of the first version of the image; and

the converting comprises,

5　　weighting groups of transform coefficients associated with respective subgroups of pixel values from the second group of pixel values, and

mathematically combining the weighted transform coefficients within each group of transform coefficients.

10　36.　The method of claim 31 wherein:

the first group of transform values represents a second group of pixel values that represents the portion of the first version of the image; and

the converting comprises,

weighting groups of transform coefficients associated with respective

15　subgroups of pixel values from the second group of pixel values, and

summing corresponding ones of the weighted transform coefficients within each group of transform coefficients.

37.　The method of claim 31 wherein:

20　each of the transform values in the first group of transform values respectively comprises a Discrete-Cosine-Transform value;

the first group of transform values represents a second group of pixel values that represents the portion of the first version of the image; and

the converting comprises,

25　weighting groups of Inverse-Discrete-Cosine-Transform coefficients associated with respective subgroups of pixel values from the second group of pixel values,

summing corresponding ones of the weighted coefficients within each group of coefficients to generate respective groups of summed coefficients,

30　and

mathematically combining the second group of transform values and the groups of summed coefficients according to an Inverse-Discrete-Cosine-Transform algorithm.

38. A method, comprising:

modifying a motion vector associated with a portion of a first version of a first image;

identifying a portion of a second image to which the modified motion vector points, the second image having a different resolution than the first version of the first image; and

generating a portion of a second version of the first image from the identified portion of the second image, the second version of the first image having the same resolution as the second image.

39. The method of claim 38 wherein the second image has a lower resolution than the first version of the first image.

40. The method of claim 38 wherein the modifying comprises modifying the motion vector from being compatible with the first version of the first image to being compatible with the second image.

41. A method, comprising:

modifying a motion vector associated with a portion of a first version of a first image to be compatible with a second image having a different resolution than the first version of the first image;

identifying a portion of the second image to which the modified motion vector points;

converting a first group of residuals that represents the portion of the first version of the first image into a second group of residuals that represents a portion of a second version of the first image, the second version having the same resolution as the second image; and

mathematically combining the second group of residuals and pixel values that represent the identified portion of the second image to generate pixel values that represent the portion of the second version of the first image.

42. The method of claim 41 wherein the second image and the second version of the first image have a lower resolution than the first version of the first image.

5
43. The method of claim 41 wherein:

the second image and the second version of the first image have a lower resolution than the first version of the first image; and

the second group of residuals has fewer residuals than the first group of residuals.

10
44. The method of claim 41 wherein the modifying comprises multiplying the motion vector with a scaling factor between the first version of the first image and the second image.

15
45. The method of claim 41 wherein the modifying comprises modifying the motion vector to have a resolution of less than ½ pixel in at least one dimension.

46. The method of claim 41, further comprising calculating the pixel values that represent the identified portion of the second image.

20
47. The method of claim 41 wherein the mathematically combining comprises summing each of the residuals in the second group with a respective one of the pixel values that represent the identified portion of the second image.

25
48. The method of claim 41, further comprising calculating the pixel values that represent the identified portion of the second image by interpolating between actual pixels of the second image.

49. The method of claim 41 wherein the converting comprises:
30
selecting a first group of transform values from a second group of transform values that represents the first group of residuals, the first group of transform values being smaller than the second group of transform values; and

converting the first group of transform values directly into the second group of residuals.

l^0

$P_{0,0}$ .... $P_{0,n}$

$P_{t,n}$ .... $P_{t,n}$

12

$P_{0,0}$ .... $P_{0,n/g}$

$P_{\frac{T}{h},0}$ .... $P_{T_h,n/g}$
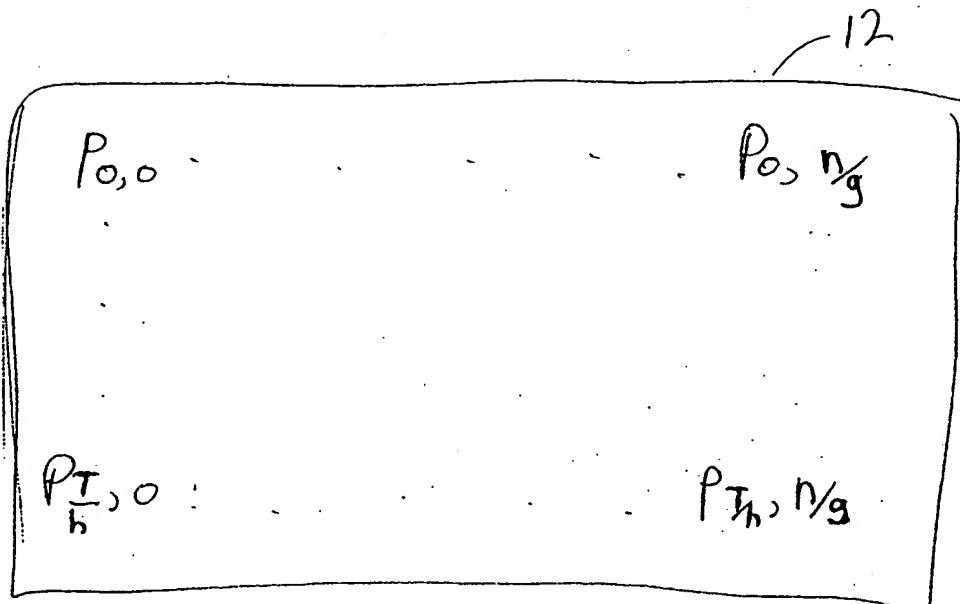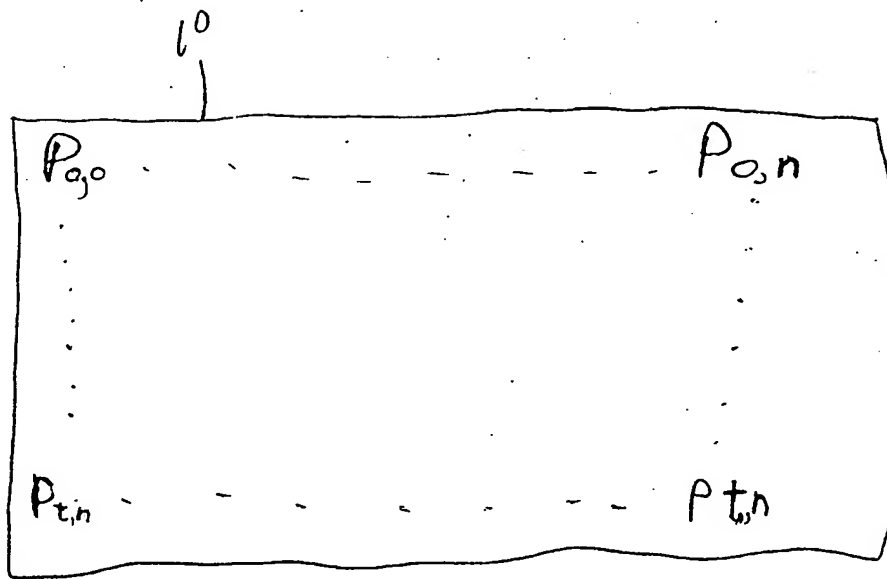
Figure 1 (prior art)

Fig 2

Fig. 3B
(Prior Art)

Y VALUES

Fig. 3A
(Prior Art)

Fig. 3C
(Prior Art)

CB VALUES

Fig. 3D
(Prior Art)

CR VALUES

Fig. 2
(Prior Art)

57

$$Y/_4 - DCT_{(0,0)a} \qquad \qquad Y - DCT_{(0,7)a}$$

$$Y - DCT_{(7,0)a} \quad \cdots \quad Y - DCT_{(7,7)a}$$
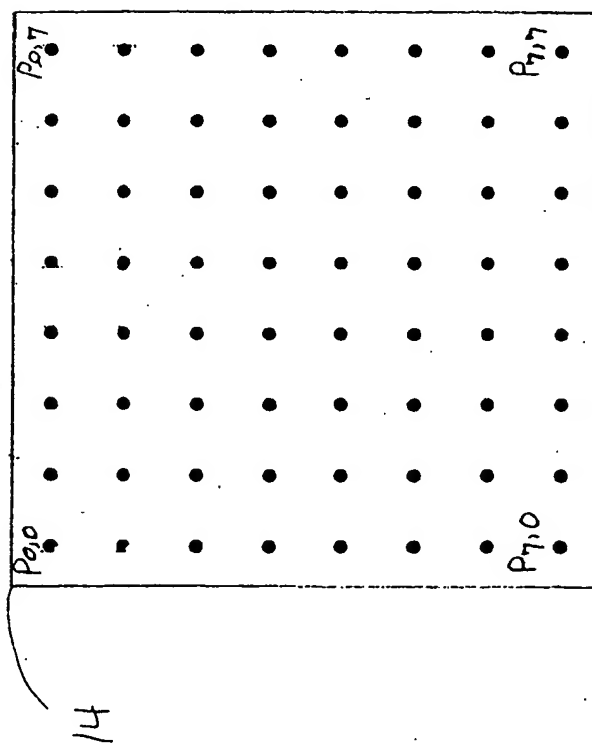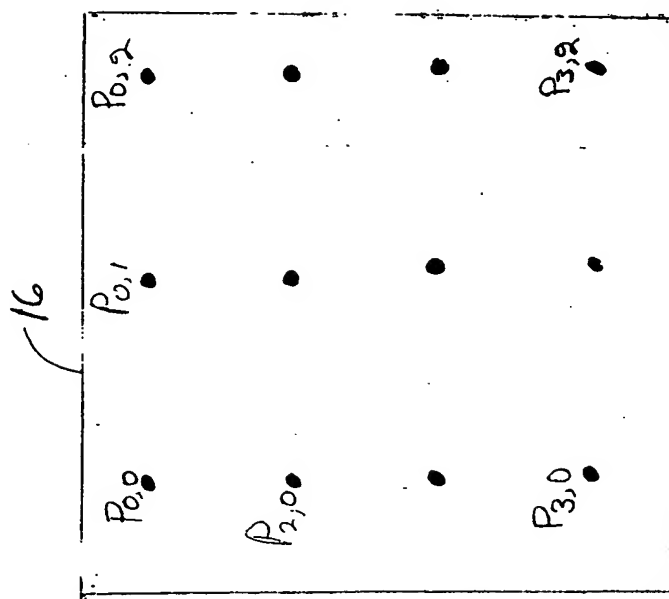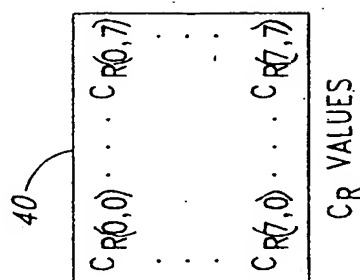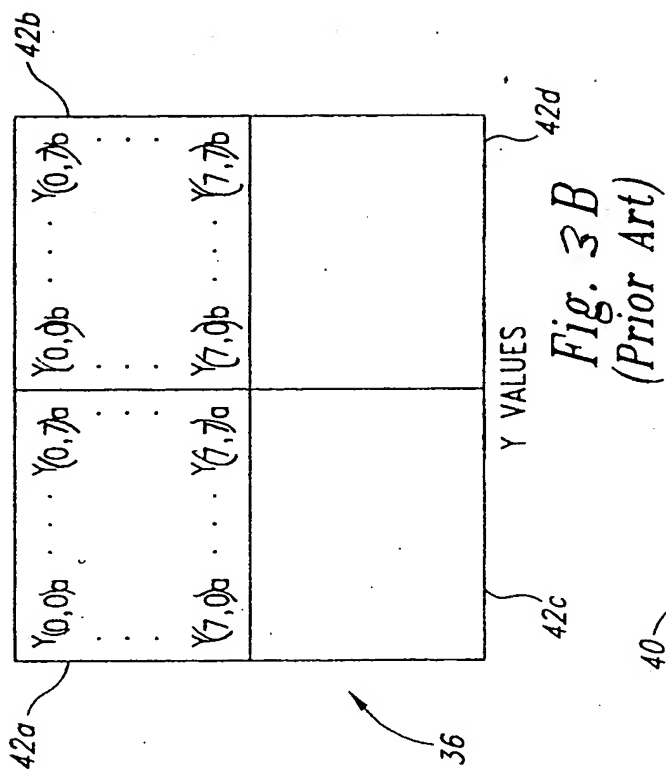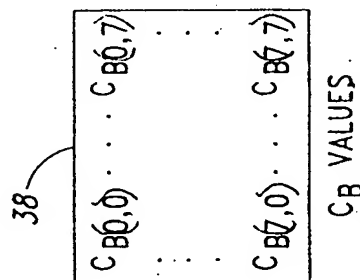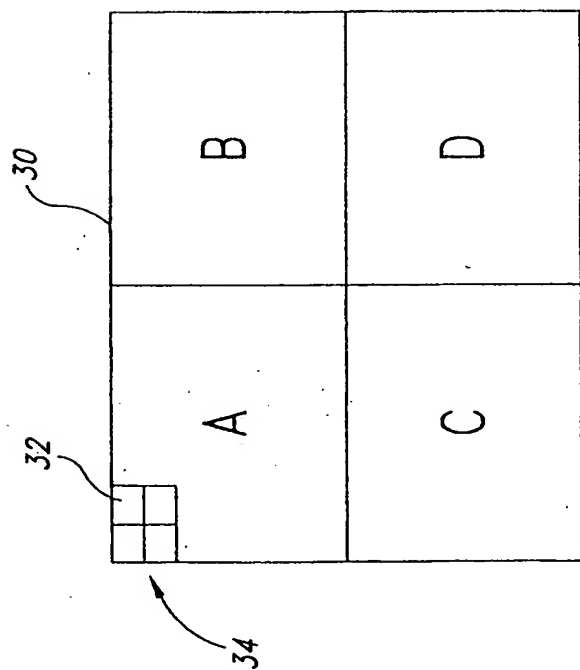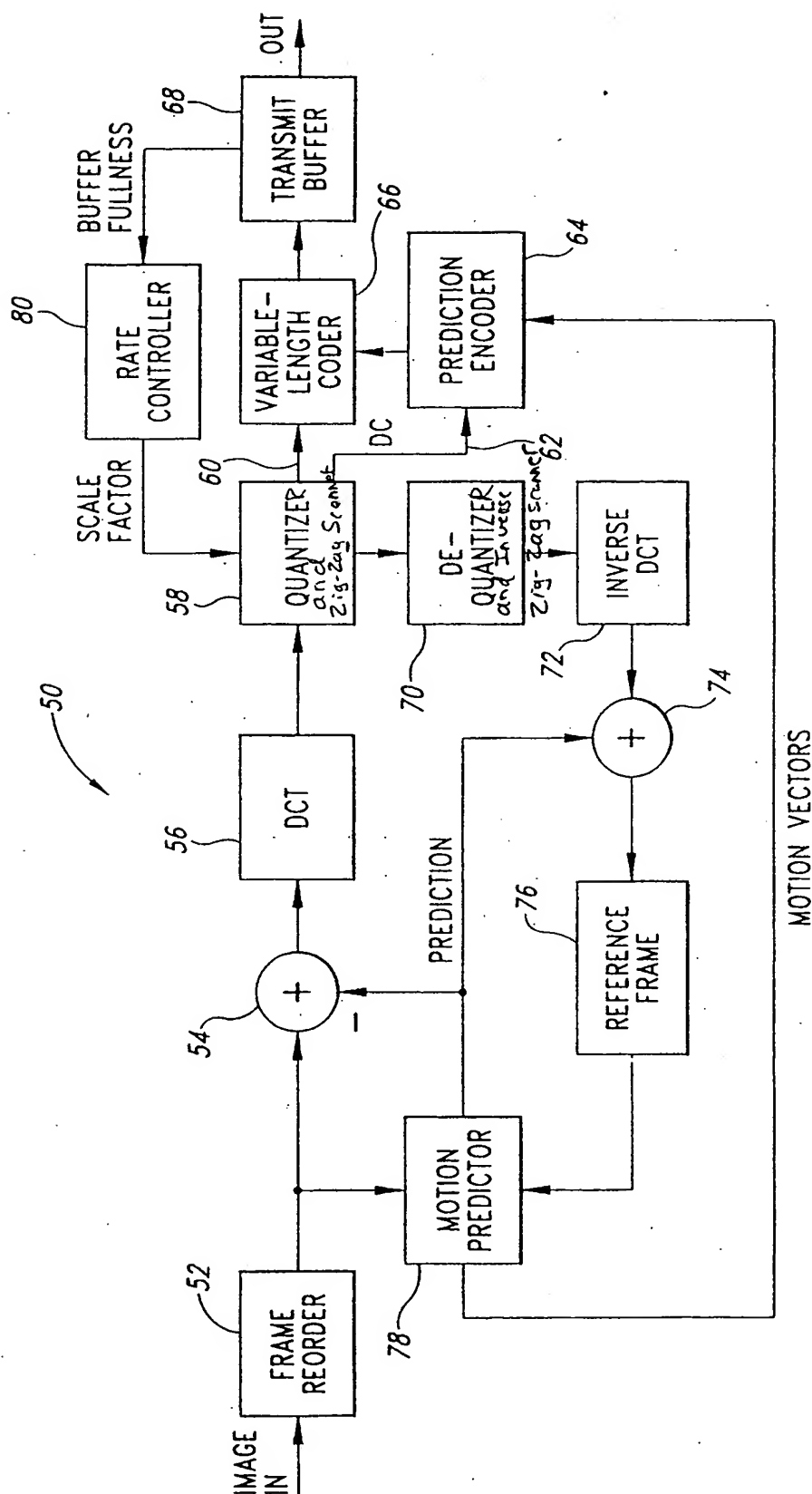
Figure 5

Figure 9

89

$$Y - IDCT_{(0,0)a} \quad \cdots \quad Y - IDCT_{(0,7)a}$$

$$Y - IDCT_{(7,0)a} \quad \cdots \quad Y - IDCT_{(7,7)a}$$
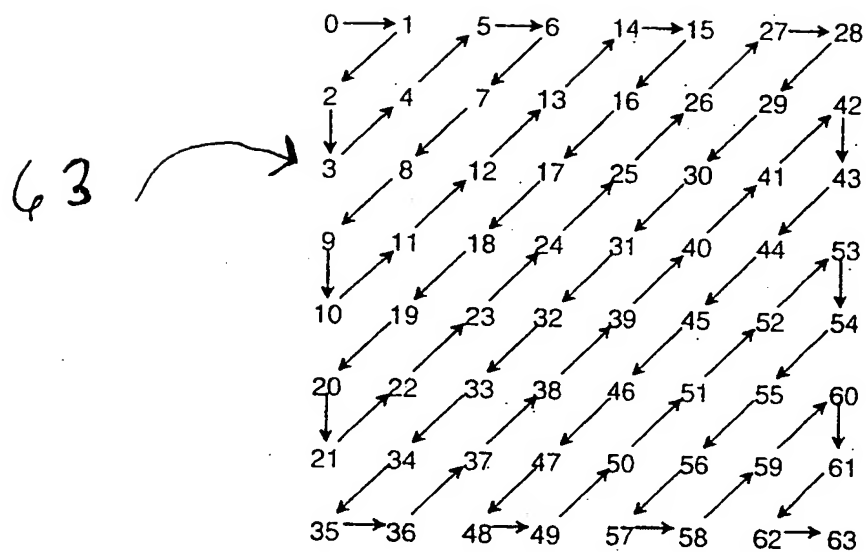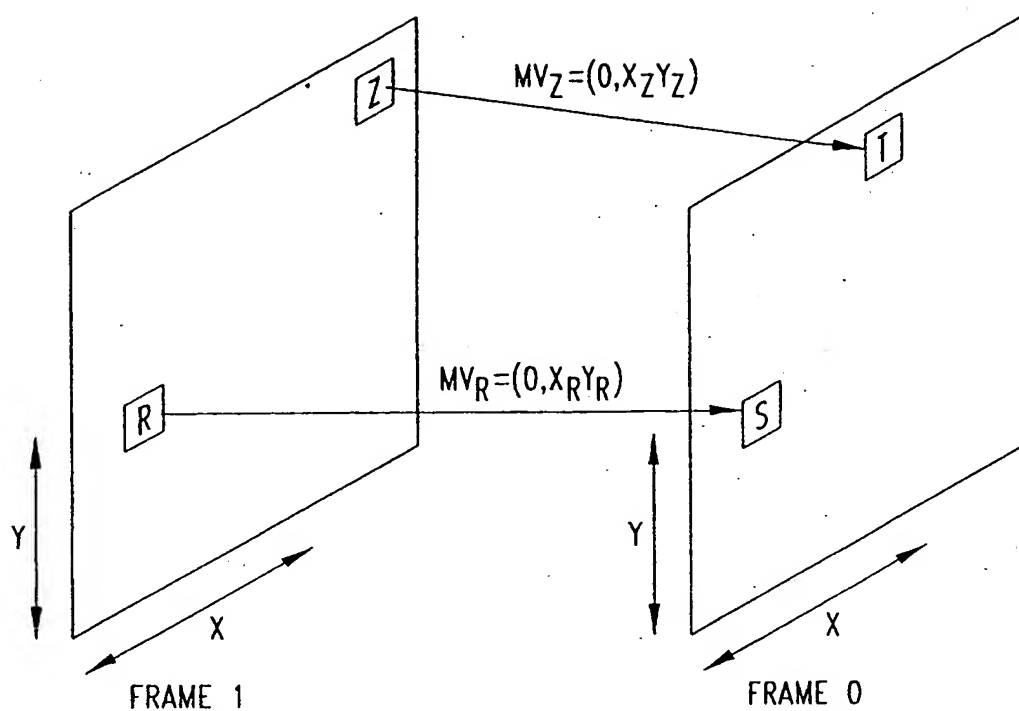
Figure 6
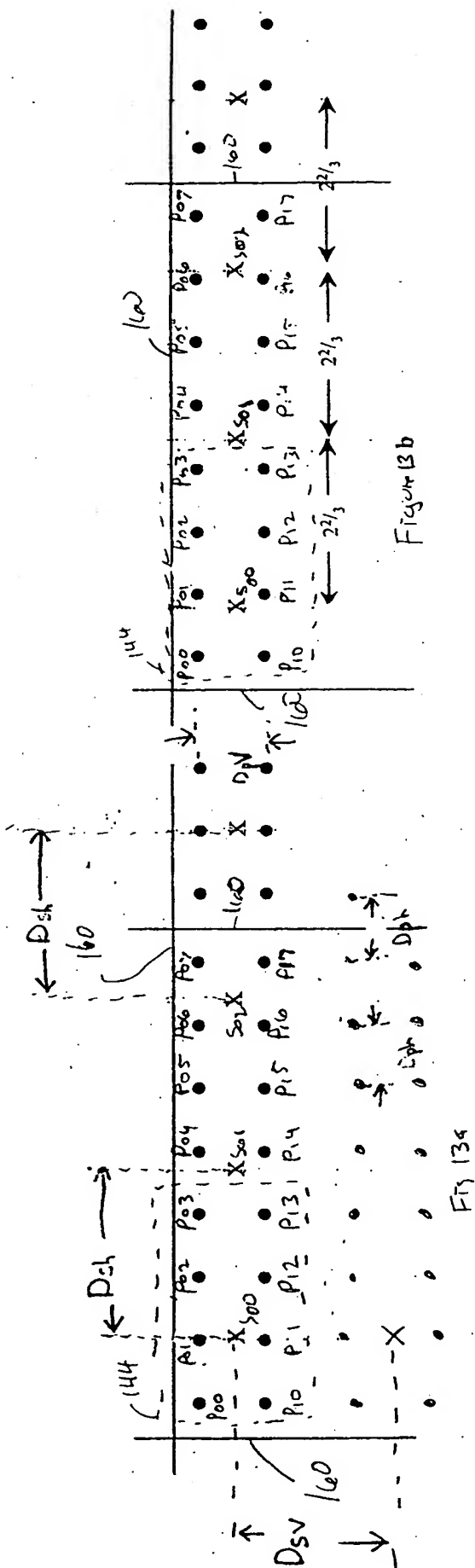
Fig. 7
(Prior Art)

Fig. 8
(Prior Art)
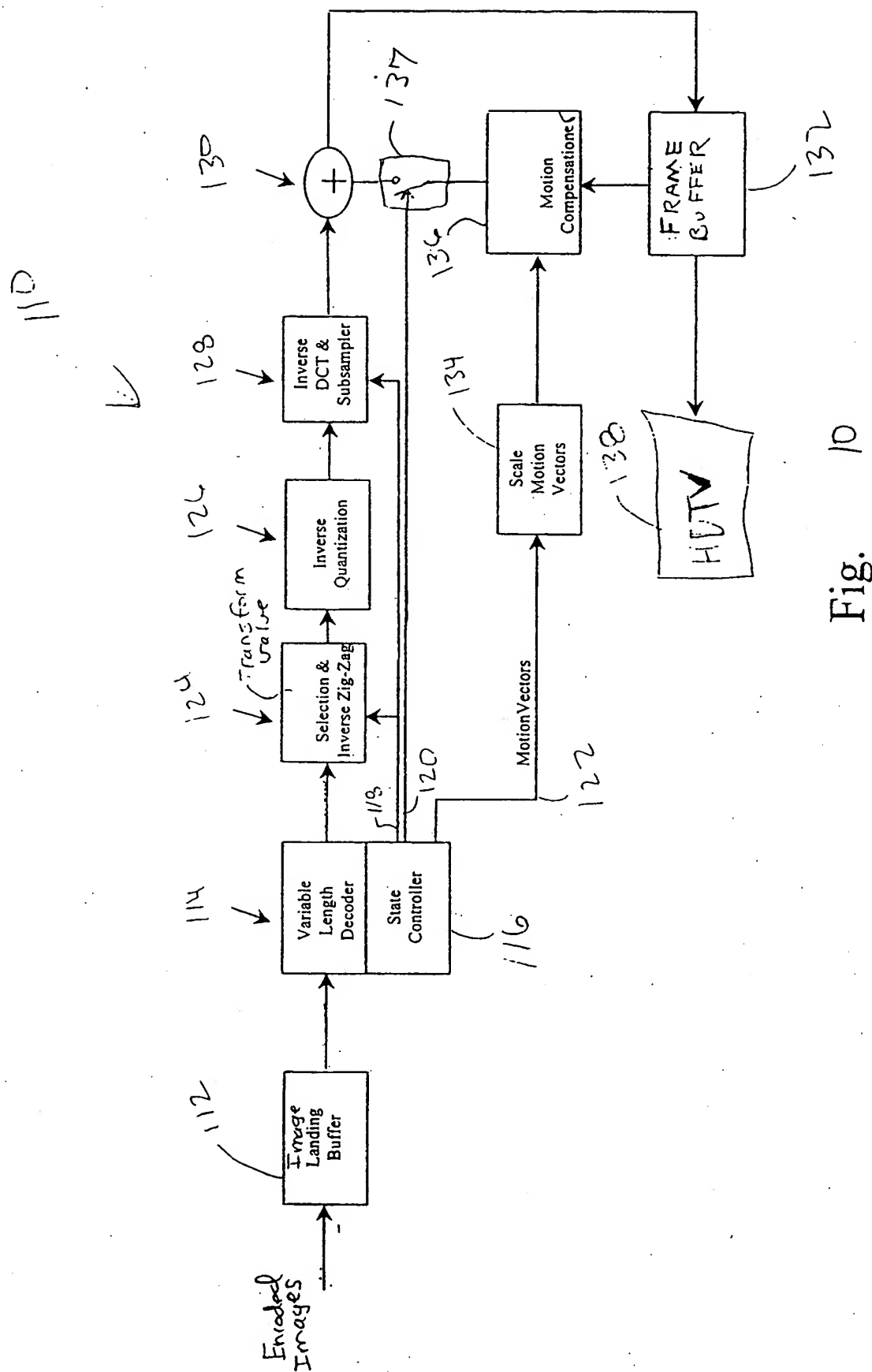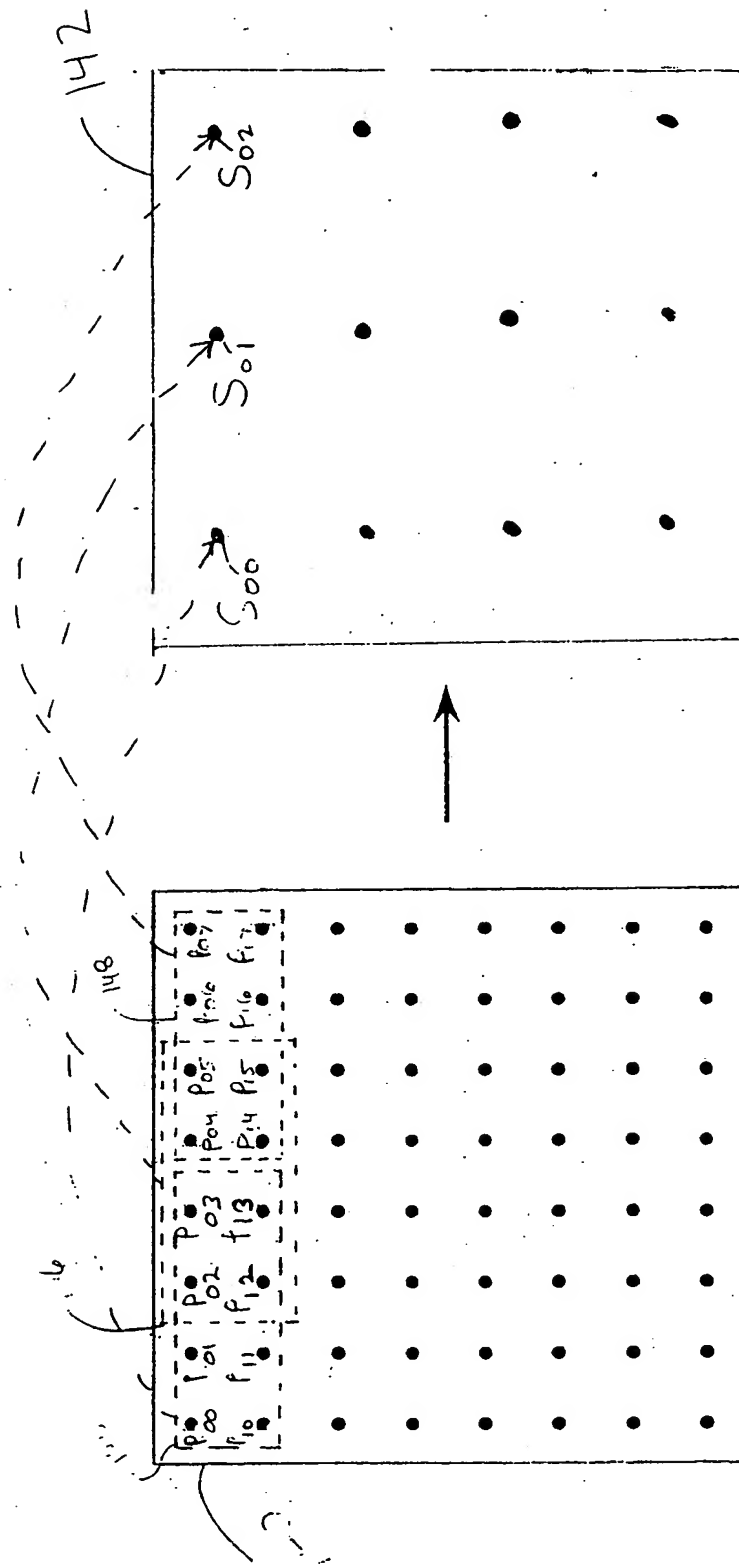
Fig. 10
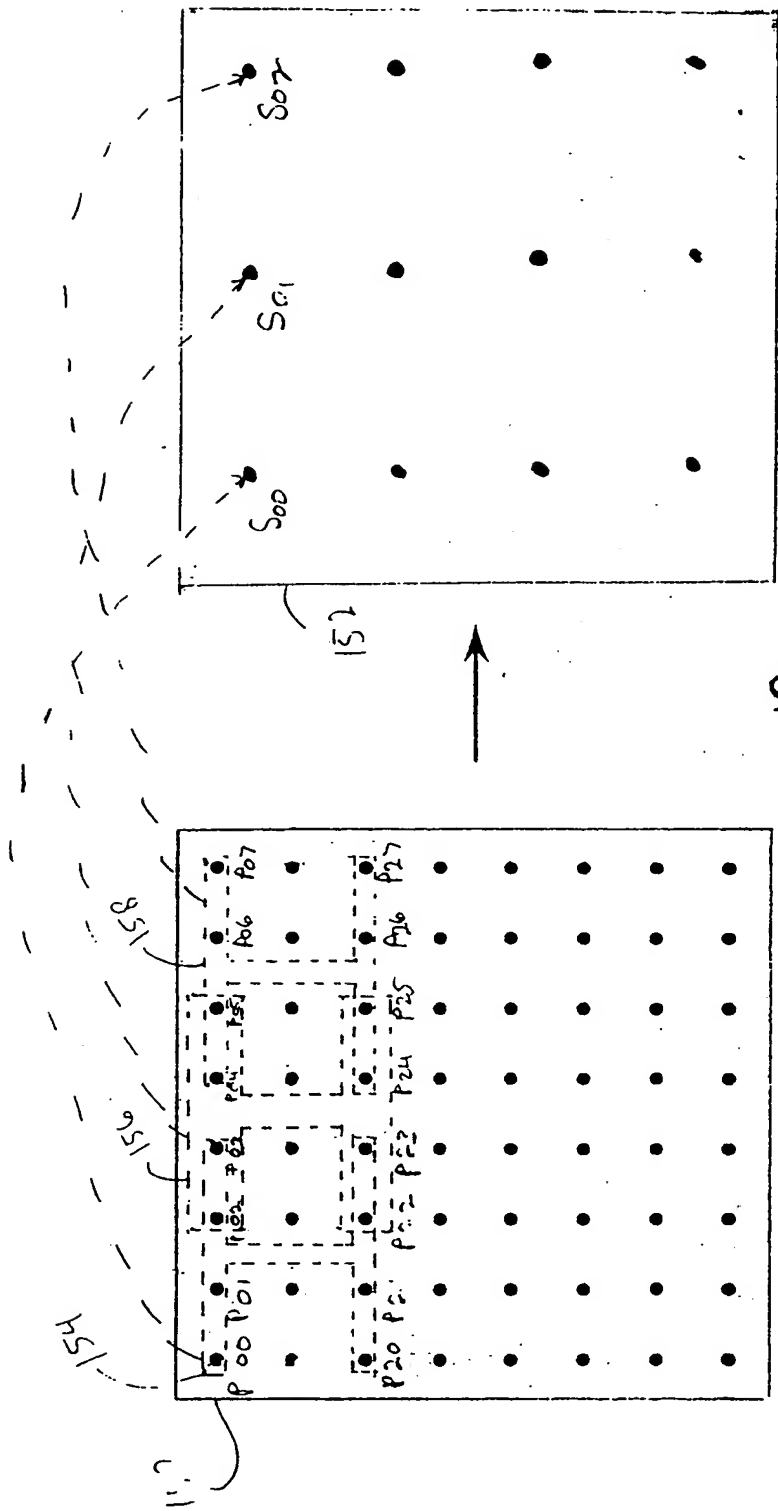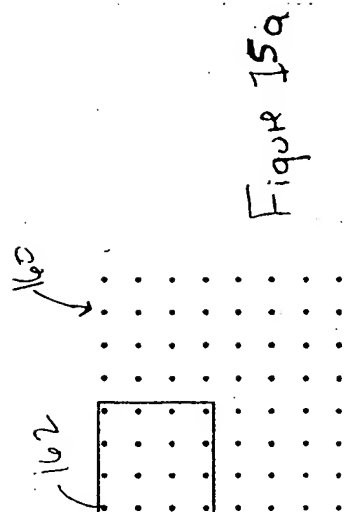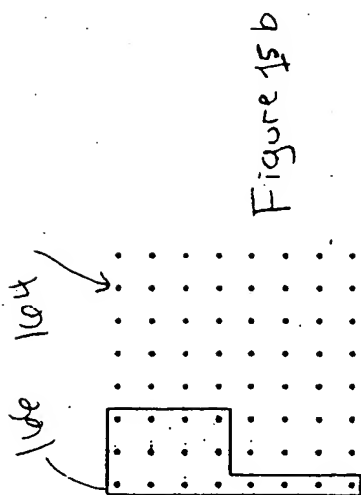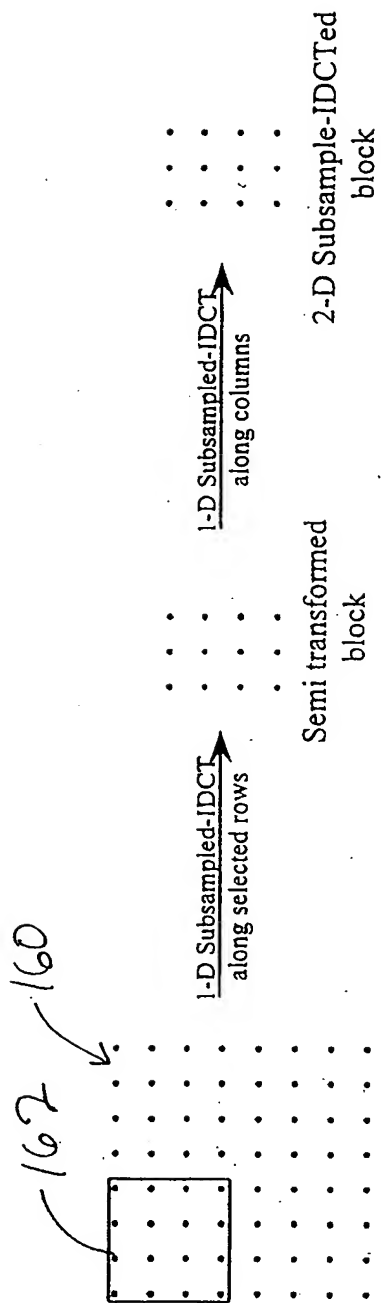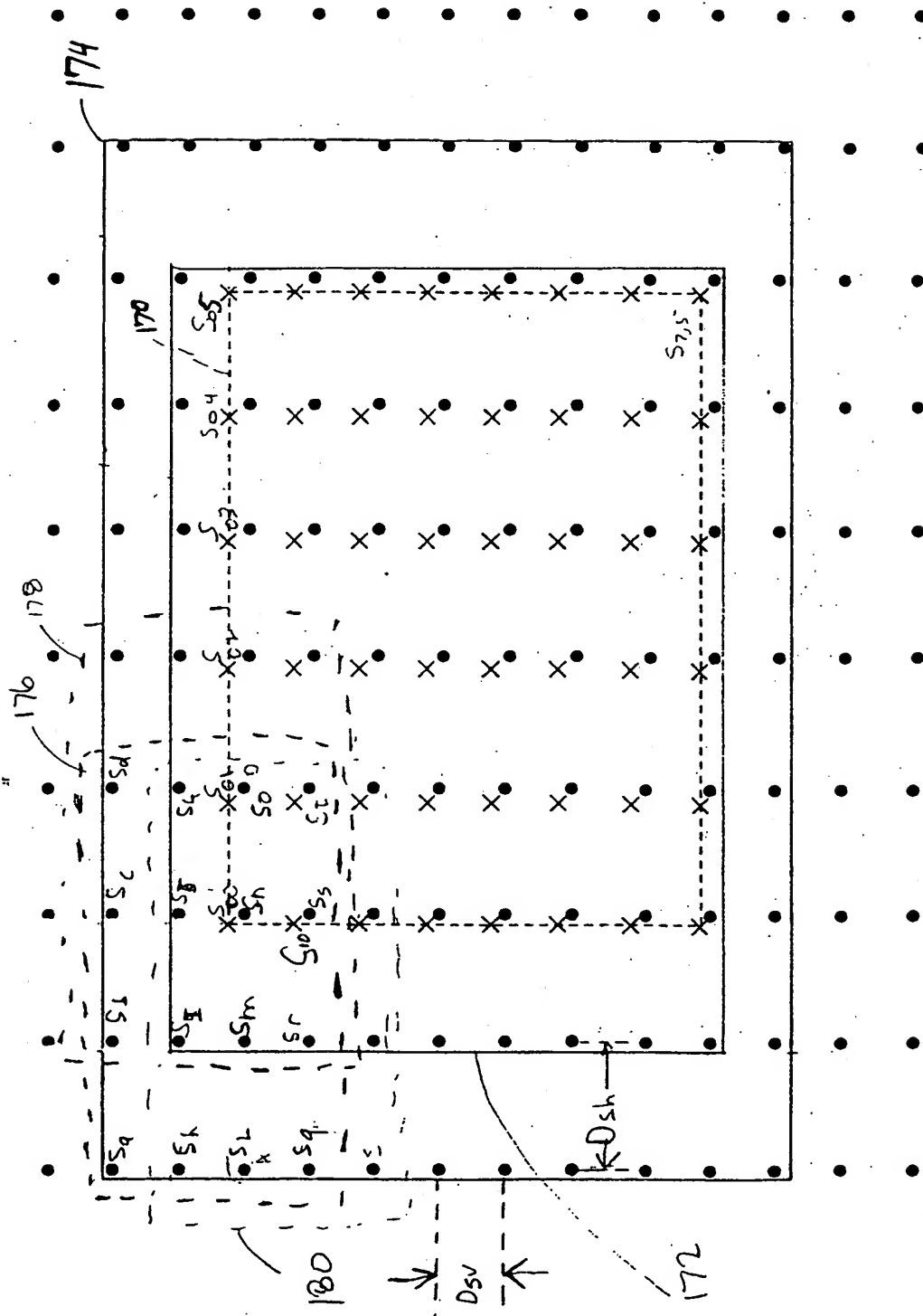
Fig. 11

Fig. 12

Figure 13 b

Fig 13a
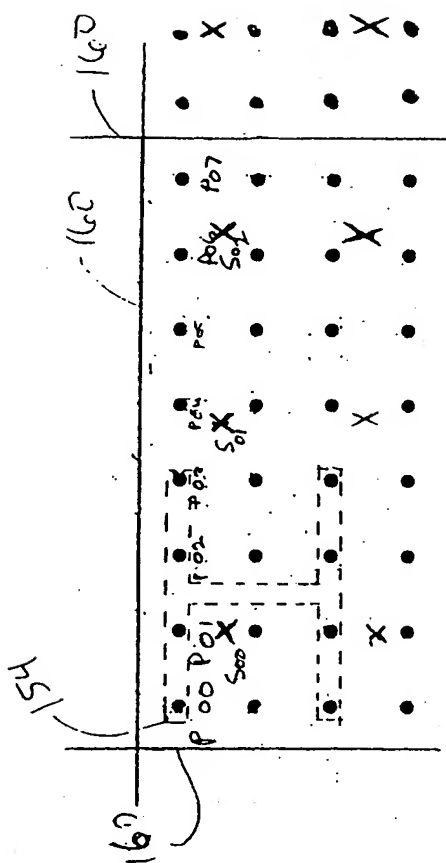
Fig. 14

Figure 15a

Figure 15b

1-D Subsampled-IDCT along selected rows

162

160

Semi transformed block

1-D Subsampled-IDCT along columns

2-D Subsample-IDCTed block

Fig. 16

Fig 17

INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06K 9/36
US CL   : 382/236, 382/250

According to International Patent Classification (IPC) or to both national classification and IPC

B.   FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. :   382/236, 250, 233, 299; 348/403-408

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C.   DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,761,343 A (HARUMA et al) 02 June 1998, column 7, line 1 through column 8 line 9, column 3 line 1 through column 4 line 30.	1-49
Y, P	US 5,809,173 A (LIU et al) 15 September 1998, column 6, lines 10-64.	1-49
A	US 5,579,412 A (ANDO) 26 November 1996, column 2, lines 23-65.	1-49

☐ Further documents are listed in the continuation of Box C.   ☐ See patent family annex.

	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family.

Date of the actual completion of the international search	Date of mailing of the international search report
21 OCTOBER 1999	**17 NOV 1999**

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Paul H. Schirduan *James R. Matthews*
Facsimile No. (703) 308-9051	Telephone No. (703) 308-7800